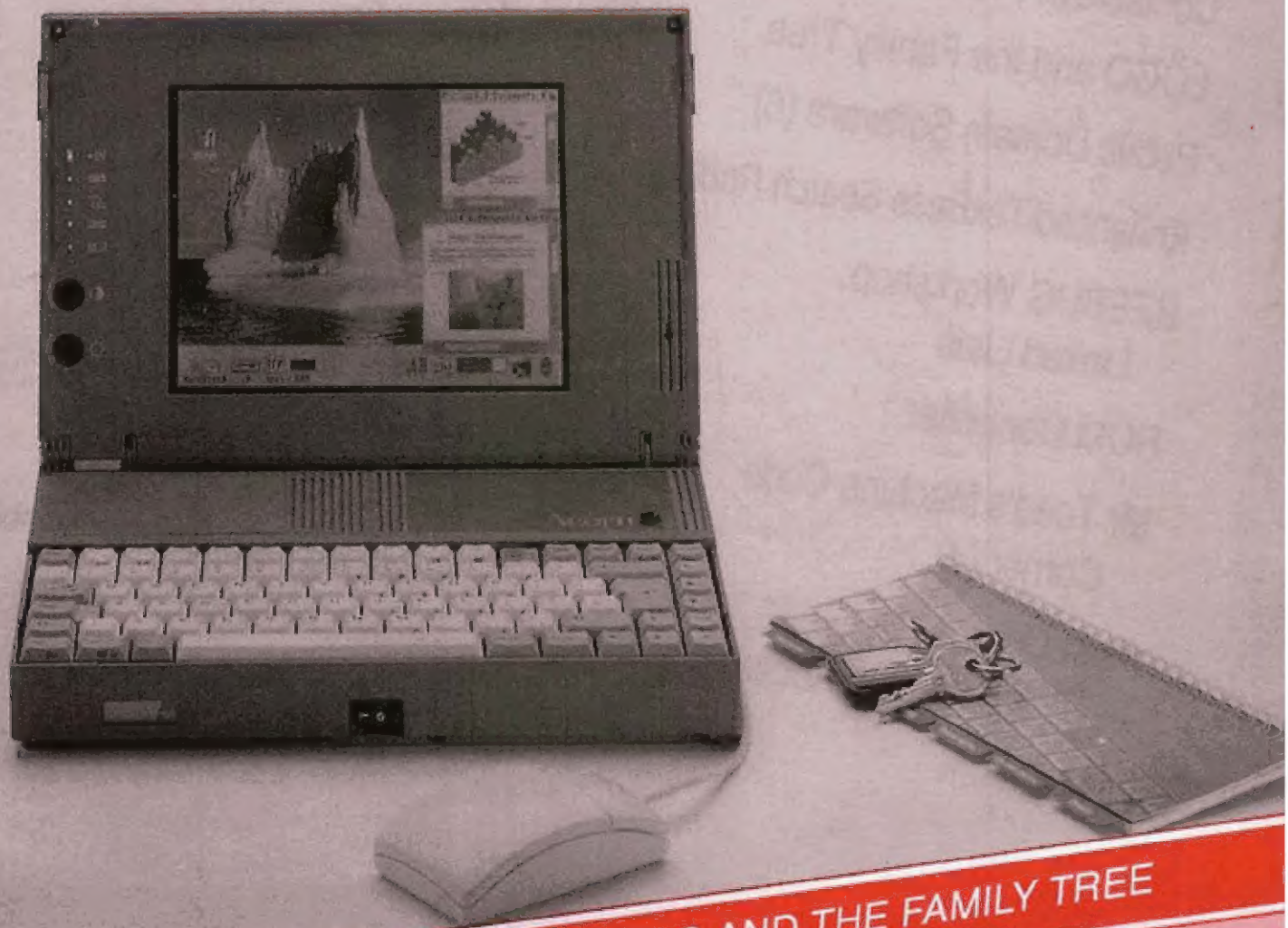
Vol.11 No.3 July 1992

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Acorn's A4 Notebook



- HIGH SECURITY CIPHERS
- LOGO AND THE FAMILY TREE
- CORKSCREW: PROCEDURE MANAGER
- ROM CONTROLLER

BEEBUG Vol.11 No.3 July 1992

FEATURES

REVIEWS

REGULAR ITEMS

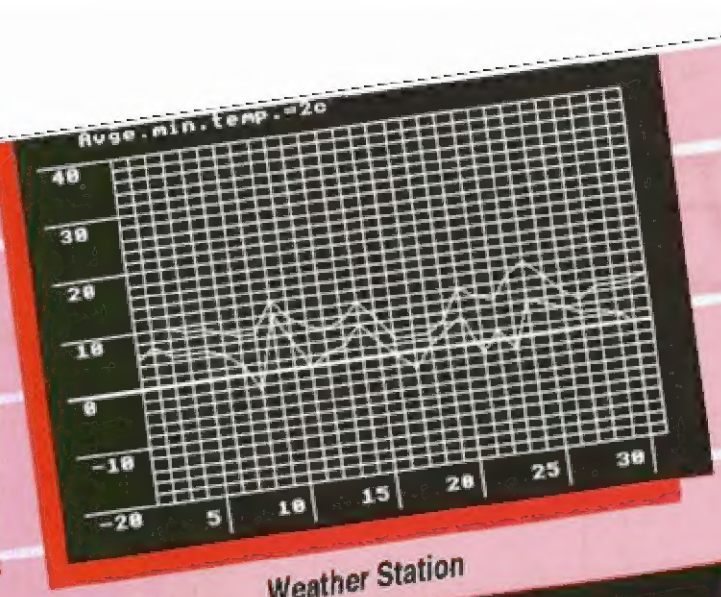
HINTS & TIPS

PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints

Acorn's A4 Notebook



Weather Station

```
Corkscrew
by Ian Palmer

Output file : ERIC
Start line  : 100
Extract : CALENDA.PROCcal

Searching B.CALENDA
Extracting PROCcal (PROCcal)
Searching B.CALENDA
Extracting FNdow (FNdow)
Searching B.CALENDA
Extracting FNmonth (FNmonth)
Searching B.CALENDA
Extracting FNvalid (FNvalid)
Searching B.CALENDA
```

Corkscrew: Procedure Manager

```
FAMILY TREE PROGRAM

The SMITH Family Tree

1. Save tree
2. Load tree
3. Create / extend tree
4. Modify existing tree
5. View generation
6. View individual's details
7. Quit

Your choice?

Free workspace 5736 1788
```

Logo and the Family Tree

```
TITLE "TexBase
1  Last month I introduced you to the idea of TexBase as a system to store and
2  retrieve text based data. If you typed in the listings given, you should
3  have been able to enter and store text. If you did enter some text, you
4  would have found that you could not actually do anything with it at the
5  time. However, we will solve that problem now by moving on to the
6  important part, the search options.
7
8
9
10
11
12
13
14
15
16
17
(N)ext  (E)xit  (P)rint  (S)ave
```

Enhanced TexBase Search Routine

```
> 15  TERMINAL
  14  VIEW
  13  Acorn ADFS
  12  BASIC
  11  Edit
  10  ViewSheet
   9  DFS
   8  Acorn ANFS 4.25
   7  RAM
   6  RAM
   5  RAM
   4  RAM
   3  RAM
   2  RAM
   1  INTER-WORD
   0  SPELLMASTER

<O>uit                    <I>nfo
<O>scli                   <S>ave
<A>ctivate     (+CTRL  = All ROMs
<D>isable      (+SHIFT = Wildcard
```

ROM Controller

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

PRODUCT REVIEWS

From time to time I receive letters from readers which seek to take me to task for the lack of reviews and other coverage of new products (including games) released for the BBC micro range. Indeed, some readers seem to hold us to blame for the lack of new product reviews. As a magazine, though, we largely reflect the state of the market; we do not create it.

Our policy has always been to carry reviews of new products, and anyone who has an extensive back issue collection will see readily what I am referring to. However, over the last two years in particular the flow of new products for the BBC micro has all but dried up as the major players devote their attention to the expanding Archimedes market.

Now I do not think it is altogether reasonable to blame these developers for the current state of affairs. First of all, developers will always look towards the future and devote their efforts where they believe they are likely to make the best return. The Archimedes (and RISC OS) is still a developing series with an expanding user base. The potential of the Archimedes is far from being exhausted and there is still much opportunity for new software (and hardware) to improve on what is already available, and to meet needs as yet unsatisfied.

This leads on to my second point. During its lifetime the capabilities of the Beeb were developed through the use of sophisticated software far beyond what was originally thought possible. Many applications of all kinds broke new ground in developing the Beeb's capabilities to their maximum. And some of the most innovative software, such as the renowned game of Elite, has been developed further for the Archimedes. More seriously, for example, Computer Concepts developed first Wordwise, then Wordwise Plus, and ultimately Interword as word processors. Indeed the latter is also available in a form suitable for the Archimedes. That's quite some achievement, which is unlikely to be bettered on the Beeb.

Thus we are left with the position that we are in today. Occasional new items do appear - for example, the educational software reviewed both this month and in Vol.11 No.1 from the Scottish Council for Educational Technology, and the Music Publisher (reviewed in BEEBUG Vol.11 No.2) from Hybrid Technology, who are still stalwarts of the BBC micro in their specialised music world.

As far as BEEBUG is concerned, we will continue to seek out and review any new products which come along. You can help in this respect by telling us of any (perhaps more obscure) new products of which you become aware. We would also welcome short contributions on readers' experiences in using major products - such information may help newer users who are unfamiliar with these established products. And if you have any further ideas for addressing this area then please let us know.

NEXT ISSUE

This is just a small reminder that the next issue of BEEBUG is the joint August/September issue, due for dispatch approximately mid-August.

M.W.

STOP PRESS.....

We have obtained details of Acorn's new laptop Archimedes just in time for this issue - see News page and full review on page 6.

ACORN ANNOUNCES ARCHIMEDES LAPTOP

Acorn Computers has announced details of its long awaited laptop computer, to be known as the *Acorn A4*, from the size of its 'footprint'. Acorn's laptop comes in two versions, one with 2Mb of RAM and 3.5" floppy disc drive, and one with 4Mb of RAM, 3.5" floppy disc drive and 60Mb IDE hard disc. In all other respects the two systems are identical.

The A4 is packaged much as other portables in the PC and Apple markets. The lid hinges up to reveal a monochrome supertwist LCD display, and a compact but full function keyboard. A substantial part of the A4's weight comes from the NiCad battery pack, and there is also a mains adaptor/charging unit included. Acorn has also chosen to supply a conventional three-button mouse with the A4.

Internally the A4 is powered by the same 25Hz ARM3 processor as used in the A5000 released last year, and with RISC OS version 3.1, the A4 is very much an A5000, but without that machine's full expansion capability.

There are connectors for parallel and RS232 serial communication, and for connecting external VGA monitor and full-size keyboard if desired.

The A4 thus represents a powerful Desktop system in a compact package. The 2Mb version costs £1399 ex. VAT (£1099 to education), and the 4Mb/60Mb version costs £1699 ex. VAT (£1399 to education). Acorn says that it expects to have A4 systems available in quantity by September/October this year.

For further information contact Acorn Computers Ltd. at Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, tel. (0223) 245200, fax (0223) 210685.

At the last minute we have managed to squeeze a comprehensive review of this exciting new development into this issue of BEEBUG. Please refer to this for further information.

ACORN ROAD SHOWS

Each year Acorn organises a number of Road Shows, when Acorn together with 25 or more third party suppliers tour the country for a week, putting on one day exhibitions aimed primarily at the educational market.

One such Road Show took place in May of this year, visiting Aberdeen, Bannockburn, Middlesbrough, West Midlands and London. A further Road Show has now been arranged from 21st to 25th September 1992 calling at Belfast, Dublin, Galway, Limerick and Cork. RISC Developments, publishers of BEEBUG, will be taking part and demonstrating all of the company's latest products. A further Road Show is also planned for England in late November.

As might be expected, the Road Shows concentrate heavily these days on the Archimedes range, but they do provide an excellent opportunity for teachers and others in education to see what is available, including Acorn's new laptop (see above).

For more information on Acorn Road Shows contact Acorn as above.

BBC ACORN USER SHOW

BBC Acorn User has announced the dates of the 1992 show, which will again be held at the Wembley Exhibition Centre. The show will be open to the public from 16th to 18th October 1992. As usual, BEEBUG magazine will be on the RISC Developments stand. More details of the show will be available nearer the time.

Acorn's A4 Notebook

Reviewed by Mike Williams

Product	**Acorn A4**	
Supplier	**Acorn Computers Ltd. Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN. Tel. (0223) 245200**	
Price	**£1399 ex. VAT (2Mb) £1699 ex. VAT (4Mb/60Mb)**	
Educational price	**£300 less in each case.**	
Optional extras	**2Mb memory upgrade**	**£110**
	60Mb hard disc upgrade	**£350**
	Additional battery pack	**£50**
	Econet interface	**£50**
	Custom made carry bag	**£35**
	A4 Technical Reference Manual	**£65**

All above prices are quoted ex. VAT

As reported on the News page, Acorn has announced details of its portable notebook computer. The *Acorn A4* is an ARM3 based system, running RISC OS 3 (version 3.10) to provide a sophisticated and powerful system in a small, compact package. The A4 will be available in two models, one with 2Mb of RAM and 3.5" floppy disc drive, and the other with 4Mb of RAM, 3.5" floppy disc drive, and 60Mb IDE hard disc drive.

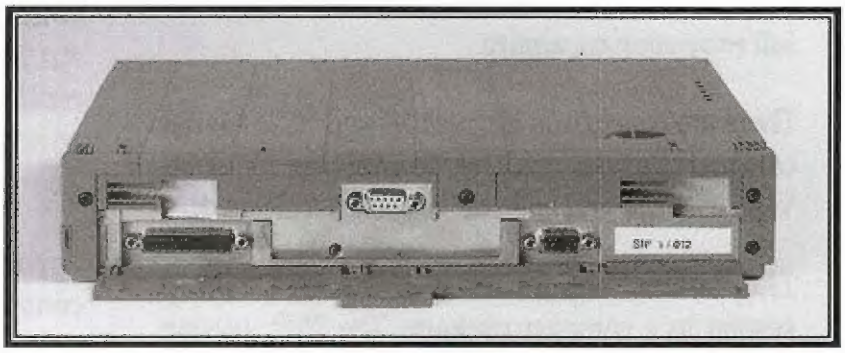
This review is based on relatively limited exposure to the new machine at Acorn's premises in Cambridge, and we are not therefore able to assess its true worth and portability as a result of genuine practical use.

INITIAL IMPRESSIONS

The polycarbonate case is simple but attractive, finished in a medium grey colour with the Acorn logo in green on top of the lid. It is 297mm by 210mm in area, and 53mm thick. The A4 is surprisingly heavy if you haven't come across other notebook computers (3kg in total), but quite a lot of that weight stems from the removable battery pack.

The front edge contains the power on/off switch and battery level indicator, while at the rear are a standard parallel port, a 9-pin D-type connector for input from the mains adaptor/charger supplied, and a further 15-pin socket for connecting an external monitor. These connectors are hidden behind a flap when not in use.

At the left, looking from the front, there is access to the slide-out battery pack, while the right-hand edge incorporates the floppy disc drive (nearer the front), and behind another flap connectors for stereo sound, an external full-size keyboard, mouse and RS232 serial port.



A4 rear connectors (parallel, PSU, external monitor)

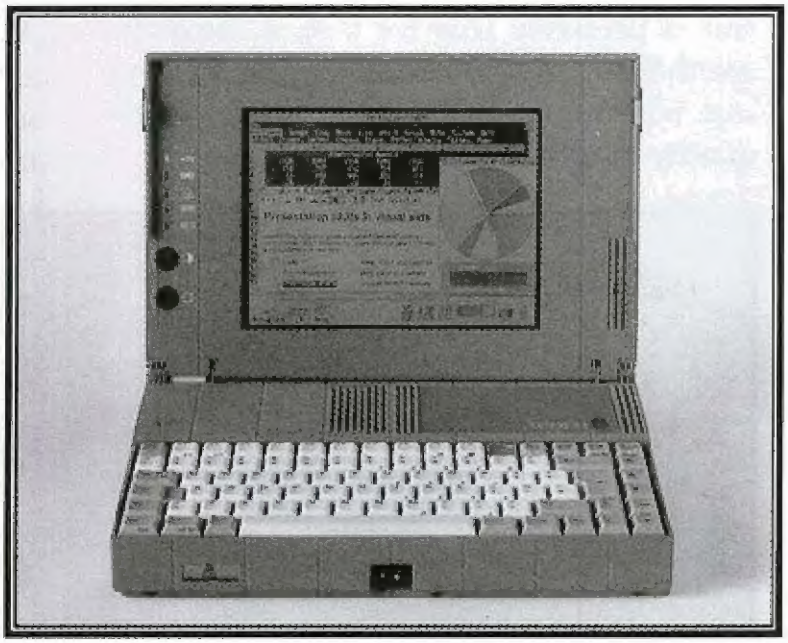
Opening up the lid reveals both the display, which is incorporated into the lid, and the keyboard. The lid also has a number of light indicators and two rotary controls for brilliance and contrast on the screen display, and a very small internal loudspeaker is fitted into the right-hand side of the lid.

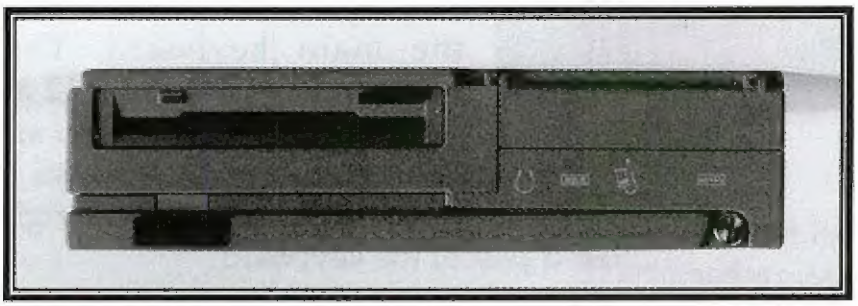
The keyboard is a full 83-key PC layout, compact as can be seen from the illustrations, and without the separate numeric keypad found on full size keyboards (though the functions of this have been simulated on the main key section). Switching on the A4 reveals the standard Archimedes Desktop display with which we are familiar. The A4 is also supplied with a full-sized mouse for controlling the Desktop environment.

Four manuals accompany the A4: as befits a RISC OS 3 machine there are the User Guide and Applications Guide as supplied with the A5000; the A4 also merits its own Welcome Guide, and in keeping with its portable nature there is a new Portable Guide to provide ready reference to key features.

SCREEN DISPLAY

The display area is 182mm wide by 137mm and comprises an edge-lit twisted LCD monochrome (black and white) display similar to that used in many other notebook style computers.





A4 floppy disc drive and remaining connectors (under lower flap)

Initial impressions are that the A4's screen is sharp, smooth and clear. Like all LCD displays, any on-screen updating leads to a lot of ghosting. In these circumstances the pointer becomes almost invisible, and dragging windows about also results (temporarily) in a confused jumble within the window being moved. This would preclude the playing of certain types of games. However, I cannot see many people buying a portable just for game playing.

In use I was not aware of any significant difference in the speed of screen refresh compared with, say, the A5000 in the office, though time limited the extent to which this could be fully tested.

The screen display operates to the PC VGA standard (mode 27 or mode 28)

Acorn has priced the two versions of the A4 at £1399 ex. VAT and £1699 ex. VAT for the 2Mb, and the 4Mb/60Mb versions respectively. Dealer fit upgrades are also available to convert the lower specification model into the higher one.

Let us look now at some of the key features in more detail, and assess the A4 in use. Remember, that this is a full-feature Desktop Archimedes running RISC OS 3, which we dealt with in our review of the A5000 in BEEBUG Vol.10 No.6.

providing a grey scale of 15 distinct shades to a resolution of 640 by 480 pixels. It is worth noting that the majority of PC notebooks and the Apple PowerBooks normally offer 32 grey shades, though Acorn claims that 15 grey shades are quite adequate for a good quality display. While this is probably true for text, it seems to me that Acorn's notebook may be missing out when it comes to displaying high quality graphics images.

A4 in use giving an indication of size

MOUSE

Acorn has chosen to include a full feature mouse as part of the A4 package, rather than go for a built-in tracker ball (as used by Apple). This is clearly a less than convenient form of control for a truly portable device. However, Desktop functions which normally require the use of a mouse can be accomplished from the keyboard as well (see below).

The mouse supplied with the A4 is to a new style. It has a much more rounded end to it, and the left and right-hand buttons are slightly sculptured to fit the fingers. It felt good to the touch, and in my view is much more stylish than the mouse currently supplied with other systems in the Archimedes range.

KEYBOARD

The keyboard supplied as part of the A4 is by necessity a compact design. Compared with the typical Beeb keyboard, that of the A4 has a more spongy feel to it, though this would no doubt become less apparent with time. As a result, the A4's keyboard is quieter, a fact which may be relevant in some uses of a portable (note taking in meetings for example). Although the QWERTY keyboard layout is (naturally) retained, not all keys are in the same positions as on other Archimedes.

There are only ten designated function keys (F1 - F10) and no numeric keypad. However, a new key, to the immediate left of the space bar, is marked 'Fn' and this operates like a Shift or Ctrl key to provide added functions in conjunction with the main keyboard. Thus function keys F11 and F12 are represented in this way, and the entire numeric keypad can also be accessed using the Fn key and the right-hand part of the keyboard.

The keyboard can also be used to emulate the mouse functions. Using Fn in conjunction with the normal cursor keys controls the movement of the mouse pointer on the screen, while the same 'shift' key in conjunction with the 'Q', 'W' and 'E' keys converts these to be equivalent to the Select, Menu and Adjust buttons of the mouse.

BATTERY POWER

The A4 comes complete with a plug-in NiCad rechargeable battery pack. The current state of the battery unit is shown in a five-segment LCD display on the

front edge of the casing. There is a Battery Management module in software which automatically monitors and controls the state of the battery unit. This will display on the icon bar a replica of the battery indicator (in a more graphical form). Battery life is estimated at between 2.5 and 3.5 hours, depending on the degree of disc activity in particular.

The Battery Manager also seeks to minimise power consumption in a number of ways. The LCD display is switched off if no keyboard activity occurs within a predetermined time. Similarly, the hard disc, if fitted, will 'spindown' to a quiescent state. Also, when nominally inactive, the clock rate of the processor is itself reduced (from 24MHz to just 6MHz), and the memory clock rate is reduced from a normal 12MHz to just 3MHz.

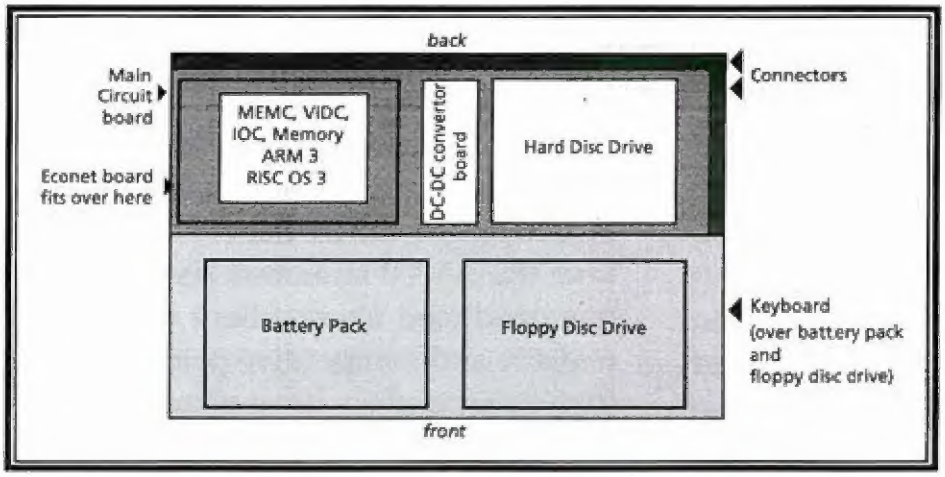


Figure 1. Layout of main features of A4

DISC DRIVES & FILING SYSTEMS

The same floppy disc drive is used in both versions of the A4. With RISC OS 3 this will support discs up to 1.6Mb using the ADFS, and up to 1.44Mb using DOS format. Thus the same range of disc formats is supported under RISC OS 3 on the A4 as on the A5000. This gives considerable flexibility, particularly if the PC Emulator is also installed on the A4.

RISC OS 3 also supports the Econet Network Filing System (NFS), and there is provision for an optional Econet interface to be fitted to the A4.

OTHER INTERFACES

Unlike other models in the Archimedes range, the A4 (by virtue of its design) is not intended to provide any expansion capability beyond the memory and hard disc upgrades from the lower spec to the higher, and Econet interface already referred to.

To my mind there is one very obvious omission in Acorn's A4 specification, and that is for an internal modem, particularly one which would have fax capability. This is by no means universal on PC notebooks, but is included in the Apple PowerBook 170, for example (though this sells at the significantly higher price of £2975). It seems to me that the ability to communicate down a telephone line is likely to be an important requirement for a good many notebook users.

HARDWARE

Any portable is bound to cram as much as possible into the available space, and Acorn's A4 is no exception (see figure 1). Most of the circuitry, including memory, MEMC, VIDC, IOC and the ARM3, is situated in the top left-hand area of the case. RISC OS itself is supplied in just two ROMs. This is all visible through a removable flap over this area.

SOFTWARE

As the A4 runs RISC OS 3, there are no obvious changes in that respect. The Applications discs, as supplied with the

A5000, are included with the lower model, but are supplied pre-installed (only) on the hard disc version.

PERFORMANCE

As already stated I could find no obvious difference in performance compared with the A5000 in the time available to me. I ran the standard Byte/PCW benchmarks (in interpreted Basic) on the A4, and the results and those for other Archimedes systems are shown in table 1. While the use of benchmarks can be a little controversial, the results at least provide a standardised comparison.

	A310		A3000	A5000		A4	
Mode	12	15	78	27	28	27	28
1.	0.32	0.42	0.28	0.07	0.08	0.07	0.07
2.	0.33	0.43	0.30	0.07	0.09	0.08	0.08
3.	1.49	1.88	1.16	0.29	0.34	0.31	0.31
4.	10.37	18.03	10.87	8.29	15.42	7.78	7.78
5.	2.01	2.51	1.89	0.47	0.53	0.47	0.51
6.	0.57	0.57	0.29	0.33	0.21	0.39	0.39

Table 1. PCW benchmark figures (in seconds) for selected machines and modes

The benchmarks test integer arithmetic (1), floating point arithmetic (2), trigonometric calculation (3), writing text to screen (4), writing graphics to screen (5) and writing to disc (6). Do not read too much into these figures - I have seen tables for similar machines and modes giving different results to those which I calculated. The A4 and A5000 give similar results except for test 4 (write text to screen) where the A4 is faster.

MARKET POSITION

Acorn says that it sees the market for the portable as being adults, and to a lesser extent, young adults. Education is one target area for the A4 but Acorn also expects strong take-up within the consumer market.

Attractive and desirable as the A4 undoubtedly is, any potential purchaser needs to ask some hard questions. The obvious one is "Do you really need a *portable* computer", because unless that is of considerable and on-going importance there is very little justification for buying one. It doesn't have colour, the smaller keyboard is less easy to use than a full sized one, and it can't be expanded. The most obvious user of any portable seems to me to be the business and professional person who travels frequently on business. A portable gives immediate access to masses of information, and enables text and figures to be entered and edited at will, wherever you may happen to be. But business people and professionals are not really noted as major users of the Archimedes (at least not yet).

Apart from any apparent provision for handling a modem and possibly the use of a mouse rather than tracker ball, there is very little to criticise as far as the A4 is concerned. Like the A5000 launched last year, Acorn has produced an excellent package at a realistic and competitive price. Let's hope that enough users have genuine need for a portable to justify Acorn's faith in such an excellent computer.

ACORN A4 PORTABLE

Beebug Ltd. as a major Acorn dealer, will be stocking both versions of the A4 notebook computer as soon as supplies are available (expected September/October in quantity).
For further information or to place an order and put your name onto Beebug's priority list contact

Beebug Ltd.,
117 Hatfield Road, St Albans, Herts AL1 4JS,
tel. 0727 40303, fax 0727 860263.

High Security Ciphers (Part 1)

Got something to hide? Ruben Hadekel explains his unique approach to encoding.

Encryption of data is not something we worry about much of the time, but when we do need it, say to protect confidential files on children in school, we want the safest system we can get. In this article I will show you how to lock data away from the most tenacious snooper.

THE PRINCIPLE

The system proposed here combines very high security with simple and fast processing; it's based on pseudo-random number series, plus four key digits, and produces no discernible pattern. The only way I can see to decode a message, even knowing the principle but not the particular keys used, is to try out possible keys until something meaningful results.

If my reasoning stands up, the odds of hitting on the right key by chance within a century are many billions to one against, assuming that a supercomputer, working 24 hours a day, can generate and evaluate a given key in one microsecond. Even if, with the progress of computer technology, the time is reduced to a picosecond, the security is still more than adequate.

The principle is quite simple: each character is enciphered by adding its ASCII code to a term in a pseudo-random series generated by a key group. The group consists of four keys, a%, b%, c%, and d%, where a% and c% can, in principle, be up to 2^31-1, i.e. over 2 billion, while b% and d% may be limited say to 33000 in the interest of reasonably fast processing, but can be higher if desired. A value of 33000 results in about 1 minute of execution time for each of b% and d% (on a Master), though any delays are avoided if the keys, and the sequence of terms that they generate, are saved to disc, as explained below.

BBC Basic allows you to define a pseudo-random sequence using a command such as X%=RND(-a%). This sequence is different for every different value of a%. Pseudo- random sequences pass all known tests for randomness, except that they repeat after a certain number of terms. In the case of the BBC RND function, that number is around 2^34, so we don't need to worry about the 'pseudo'.

If a% can have a' different values, we can define a' random sequences, all different. If in any given sequence we discard the first b% terms, and b% can have say b' different values, this gives us a total of a'b' different sequences.

If we now define another sequence by X%=RND(-c%), and discard the first d% terms in that, if c% and d% can have c' and d' different values, this gives c'd' different sequences.

If we now add corresponding terms in the a%, b% and c%, d% sequences, this results in a total number of different sequences given by a'b'c'd'/2. The divisor 2 is necessary to allow for the fact that interchanging the first and second sequence gives the same result.

With a' and c' at say 10^9, and b' and d' at 33000, this gives a figure of 5.44*10^26 possible sequences.

Having assigned values to a% and b%, an actual sequence of terms (up to 1000) is established by repeatedly using RND(1000). The number 1000 is quite arbitrary, but any attempt at still further security by using arguments greater than 1000 is largely illusory, since there are clear similarities in the sequences generated by different arguments. This is

illustrated in the Sphinx program, given below, where the process of discarding the first d% terms in the c% sequence is executed by line 260, and the actual sequence used is then generated in line 270. Line 280 stores the terms of the c% sequence in high byte/low byte order, this being more efficient than direct storage as an integer variable, which would take four bytes instead of two.

Line 320 discards the first b% terms in the a% sequence. The sequence is then generated in line 330, where it is also added term by term to the c% sequence.

Line 340 introduces an extra security element, described later. Line 350 stores the sequence resulting from the addition in the same locations (starting at U%) as were used for storing the c% sequence.

The 'clear' or 'plain text' message is entered in the *PROCenter(x%)* procedure, x% being 1 for the enciphering routine. The ASCII code for each character is stored in locations beginning at &4A38.

HIDING IT

The message is enciphered by the *PROCenciph* procedure. In the Sphinx program, line 1530 rejects any non-alphabetic character, and line 1540 converts all characters to upper case. Line 1560 in the Sphinx program retrieves the term in the enciphering sequence, and line 1570 adds it to the ASCII code for the plain text character, while in that line MOD 26 + 65 converts the result into an ASCII code that will yield the enciphered character. Line 1600 takes care of errors resulting in blocks of less or more than five letters.

FINDING IT AGAIN

The deciphering process is more or less the converse of enciphering, but a little more tricky. The basic data are the term in the enciphering sequence (T% - line 1780 in the Sphinx program) and the enciphered character G% (line 1710 in the Sphinx program). We also know that these are related by the enciphering line 130, so that the variable S% (line 1790) must satisfy the equation:

```
G%=(C%+T%) MOD 26+65=(C%+T%)-S%+65
```

where C%, the code for the plain text character, is c+65, c being the position of the character in the alphabet. Hence:

```
S%/26 = (130+T%+G%)/26 + c/26
```

and since S%/26 must be an integer, while c/26<1, we can write:

```
S%=130+T%+G%
```

which leads to the solution expressed in lines 1800, 1810 and 1820 in the Sphinx program.

There are 3.15*10^13 microseconds in a year. If a supercomputer can evaluate a tentative key group (i.e. sequence) in one microsecond, the probability of a chance hit on the actual sequence within a century is: 3.15*10^13*100/5.44*10^26 i.e. less than 10^-11.

Few persons or organisations face an opponent that can command the full-time services of a supercomputer, and therefore the above degree of security may be over the top for perhaps most potential users. The (quite modest) execution time implied by the 'discard' keys b% and d% can be avoided, and the overall key group simplified, simply by setting b% and d% to unity. In that case, assuming one millisecond for evaluating a potential sequence, the probability of a chance hit within one year is about 6*10^-8.

The one program serves both for enciphering and deciphering. The program allows for messages up to 4000 characters long - a little over 500 words.

Generating the sequence of enciphering or deciphering terms can take up to about 3 minutes on a Master series machine - longer on a BBC B. The four keys can be saved to the program disc if security arrangements are considered adequate. If they are saved, the sequence

of enciphering or deciphering terms is saved as well, thus shortening execution time, and the saved keys are displayed at the next program run, with an option to change them. Keys of one billion or over are displayed in exponent notation, with the least significant digit lost on the display.

There is an option to type in the message directly, or to use a message generated beforehand on a word processor, and saved in pure ASCII format under the name *clear*. If the message is typed in directly, there is no possibility of editing, apart from using the Delete key, which works normally. In that case do not attempt to create any paragraphs or blank lines. Also, if a word processor is not used, the message must be terminated by the '|' character.

Note that line 390 in the Sphinx program fills the area of memory used by plain text and cipher with the '|' character prior to entry. Hence if word processor text is used, it is automatically terminated by that character. If text is typed in directly, the '|' or '~' character must still be typed in to signify the end of the message to the program.

The enciphered text can be sent to the recipient(s) on disc, or as print-out, or both, with provisions for multiple copies. Just follow the prompts.

If the enciphered message is received as a print-out, it must be entered via the keyboard. There are options to type it directly (with the same rules as above, including terminating with the '|' character), or to use a word processor (before running the program), in which case it must be spooled under the name crypt. Either way, do not omit the spaces between the 5-letter blocks.

Entry errors such as more than one space between 5-letter blocks, or blocks with less or more than 5 letters, are catered for without affecting deciphering except locally. Spaces between blocks are not preserved in the decrypt in the Sphinx program, as they would only confuse the issue.

The deciphered text is shown on the screen, and that may be good enough for short messages particularly in the Sesame program (see below). Longer messages can be printed, or loaded to a word processor under the name *decrypt*, and processed to put in spaces in their proper places (for the Sphinx version), and/or to be viewed on screen at leisure. Again, follow the prompts.

OPEN SESAME

The above is a description of the Sphinx program for the most part. Sesame is virtually identical to Sphinx except that spaces are restored in their correct places in the decrypt, although the enciphered message still retains the standard format of five-letter blocks. The effect is achieved at the cost of losing the letter J. Multiple spaces are enciphered as a single space.

If any j or J is left as such in the message, it will be replaced by a space in the decrypt, but there is nothing to stop you replacing any j by i, y or g (as appropriate) in the message - nobody could misunderstand 'yustice' or 'gesuit', especially when in context.

As far as I can see, the trick used, i.e. replacing any space by J when enciphering, and any J by a space when deciphering, does not detract from the degree of security of the cipher. The result, however, is a decrypt that is easily readable without puzzlement, or further processing.

Next month we will look at ways of refining the encryption and decryption processes further to preserve the original message more exactly.

```
  10 REM Program Sphinx
  20 REM Version B 1.0
  30 REM Author  Ruben Hadekel
  40 REM BEEBUG  July 1992
  50 REM Program subject to copyright
  60 :
 100 DIMA$(2):sq%=0
 110 ON ERROR GOTO 180
 120 J%=OPENIN("savekey")
 130 INPUT#J%,a%,b%,c%,d%:ON ERROR OFF:
CLOSE#J%
 140 CLS:PRINT"Stored keys are":PRINTa%
'b%'c%'d%;''"Do these stand (Y/N)?"
 150 IF FNYN="N" THEN 180
 160 *LOAD seq 5BCC
 170 GOTO 390
 180 ON ERROR OFF:J%=OPENOUT("savekey")
 190 CLS:INPUT"Enter keys a,b,c,d "a%,b
%,c%,d%
 200 CLS:PRINT"Do you want to save the
keys to disc? (Y/N)":IF FNYN="N" sq%=1:
GOTO 220 ELSE PTR#J%=0
 210 PRINT#J%,a%,b%,c%,d%
 220 CLOSE#J%
 230 CLS:PRINT"Wait for next prompt"
 240 U%=&5BCA
 250 X%=RND(-c%)
 260 FOR X%=1 TO d%:Y%=RND(1000):NEXT
 270 FOR X%=1 TO 4000:Y%=RND(1000):U%=U
%+2
 280 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
 290 NEXT X%
 300 U%=&5BCA
 310 X%=RND(-a%)
 320 FOR X%=1 TO b%:Y%=RND(1000):NEXT
 330 FOR X%=1 TO 4000:U%=U%+2:Y%=(?U%)*
256+?(U%+1):Y%=Y%+RND(1000)
 340 IF Y% MOD 26=13 Y%=Y%+1
 350 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
 360 NEXT X%
 370 IF sq%=1 GOTO 390
 380 *SAVE seq 5BCC 7B0B
 390 FOR X%=14500 TO 23499:?X%=124:NEXT
 400 CLS:PRINT"1. Encipher"''"2. Deciphe
r"'''"Select by keying 1 or 2"
 410 G$=GET$:IF G$<>"1" AND G$<>"2" GOT
O 410
 420 IF G$="1" PROCenciph ELSE PROCdeci
ph
 430 END
 440 :
1000 DEF PROCenter(x%)
1010 CLS:PRINT"Is message on disc? (Y/N
)"
1020 IF FNYN="N" GOTO 1100
1030 CLS:PRINT"Insert disc, then press
any key":G=GET
1040 IF x%=2 GOTO 1070
1050 *LOAD clear 4A38
1060 ENDPROC
1070 *LOAD crypt 4A38
1080 ENDPROC
1090 :
1100 CLS:PRINT"Type message, ending wit
h |":V%=&4A37
1110 REPEAT
1120 V%=V%+1:G%=GET:IF G%=127 V%=V%-2 E
LSE ?V%=G%
1130 PRINTCHR$(G%);
1140 UNTIL G%=124
1150 ENDPROC
1160 :
1170 DEF PROCoutput(x%)
1180 A$(1)="encrypt":A$(2)="decrypt"
1190 CLS:PRINT"Save "A$(x%)" to disc? (
Y/N)"
1200 IF x%=2 PRINT"otherwise print"
1210 IF FNYN="N" GOTO 1330
1220 IF x%=2 N%=1:GOTO 1240
1230 INPUT"How many copies? "N%
1240 FOR M%=1 TO N%
1250 PRINT"Insert disc, then press any
key":G%=GET
1260 IF x%=1 F%=OPENOUT"crypt" ELSE F%=
OPENOUT"decrypt"
1270 W%=&3A97
1280 REPEAT
1290 W%=W%+1
1300 BPUT#F%,?W%
1310 UNTIL ?W%=124
1320 CLOSE#F%:NEXT M%
1330 CLS:IF x%=2 GOTO 1350 ELSE PRINT"P
rint-out required? (Y/N)"
1340 IF FNYN="N" ENDPROC
```

```
 1350 IF x%=2 N%=1:GOTO 1370
 1360 INPUT"How many copies "N%
 1370 VDU2:FOR M%=1 TO N%
 1380 W%=&3A97
 1390 REPEAT
 1400 W%=W%+1:PRINTCHR$(?W%);
 1410 UNTIL ?W%=124
 1420 CLS:PRINT"Paper - then press any k
ey":G=GET
 1430 NEXT M%
 1440 VDU3
 1450 ENDPROC
 1460 :
 1470 DEF PROCenciph
 1480 PROCenter(1)
 1490 U%=&5BCA:W%=&3A97:X%=0:V%=&4A37
 1500 REPEAT
 1510 V%=V%+1:G%=?V%
 1520 IF G%=124 GOTO 1610
 1530 IF G%<65 OR G%>122 OR (G%>90 AND G
%<97) GOTO 1510
 1540 G%=G% AND 223
 1550 U%=U%+2
 1560 T%=256*(?U%)+?(U%+1)
 1570 H%=(G%+T%) MOD 26 +65
 1580 W%=W%+1:?W%=H%
 1590 X%=X%+1
 1600 IF X%=5 H%=32:W%=W%+1:?W%=H%:X%=0
 1610 UNTIL G%=124
 1620 PROCoutput(1)
 1630 ENDPROC
 1640 :
 1650 DEF PROCdeciph
 1660 PROCenter(2)
 1670 U%=&5BCA:V%=&4A37:W%=&3A97
 1680 Z%=U%
 1690 PRINT"Press Shift to advance displ
ay.":PRINT:VDU14
 1700 REPEAT
 1710 V%=V%+1:G%=?V%
 1720 IF G%=124 GOTO 1840
 1730 IF G%<>32 GOTO 1770
 1740 U%=U%+10:Z%=U%
 1750 V%=V%+1:G%=?V%:IF G%=32 GOTO 1750
 1760 IF G%=124 GOTO 1840
 1770 Z%=Z%+2
 1780 T%=256*(?Z%)+?(Z%+1)
 1790 S%=130+T%-G%
 1800 IF S% MOD 26=0 H%=S% DIV 26 ELSE H
%=S% DIV 26 +1
 1810 H%=26*H%+G%-65-T%
 1820 PRINTCHR$(H%);
 1830 W%=W%+1:?W%=H%
 1840 UNTIL G%=124
 1850 PRINT:PRINT:PRINT"Press space bar
to stop here, or any "'"other key to con
tinue":G%=GET:IF G%=32 END
 1860 PROCoutput(2)
 1870 ENDPROC
 1880 :
 1890 DEF FNYN
 1900 K%=GET AND 223:IF K%<>78 AND K%<>8
9 THEN 1900 ELSE K$=CHR$(K%):=K$
```

```
   10 REM Program Sesame
   20 REM Version B 1.0
   30 REM Author  Ruben Hadekel
   40 REM BEEBUG  July 1992
   50 REM Program subject to copyright
   60 :
  100 DIMA$(2):sq%=0
  110 ON ERROR GOTO 180
  120 J%=OPENIN("savekey")
  130 INPUT#J%,a%,b%,c%,d%:ON ERROR OFF:
CLOSE#0
  140 CLS:PRINT"Stored keys are":PRINTa%
'b%'c%'d%;'"Do these stand (Y/N)?"
  150 IF FNYN-"N" THEN 180
  160 *LOAD seq 5BCC
  170 GOTO 390
  180 ON ERROR OFF:J%=OPENOUT("savekey")
  190 CLS:INPUT"Enter keys a,b,c,d "a%,b
%,c%,d%
  200 CLS:PRINT"Do you want to save the
keys to disc? (Y/N)":IF FNYN="N" sq%=1:
GOTO 220 ELSE PTR#J%=0
  210 PRINT#J%,a%,b%,c%,d%
  220 CLOSE#0
  230 CLS:PRINT"Wait for next prompt"
  240 U%=&5BCA
  250 X%=RND(-c%)
  260 FOR X%=1 TO d%:Y%=RND(1000):NEXT
  270 FOR X%=1 TO 4000:Y%=RND(1000):U%=U
```

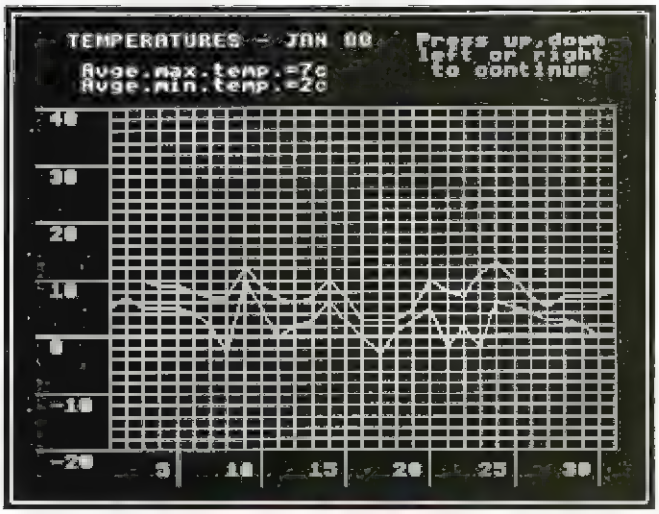
```
%+2
  280 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
  290 NEXT X%
  300 U%=&5BCA
  310 X%=RND(-a%)
  320 FOR X%=1 TO b%:Y%=RND(1000):NEXT
  330 FOR X%=1 TO 4000:U%=U%+2:Y%=(?U%)*
256+?(U%+1):Y%=Y%+RND(1000)
  340 IF Y% MOD 26=13 Y%=Y%+1
  350 ?U%=Y% DIV 256:?(U%+1)=Y% MOD 256
  360 NEXT X%
  370 IF sq%=1 GOTO 390
  380 *SAVE seq 5BCC 7B0B
  390 FOR X%=14500 TO 23499:?X%=124:NEXT
  400 CLS:PRINT"1. Encipher"''"2. Deciphe
r"''"Select by keying 1 or 2"
  410 G$=GET$:IF G$<>"1" AND G$<>"2" GOT
O 410
  420 IF G$="1" PROCenciph ELSE PROCdeci
ph
  430 END
  440 :
 1000 DEF PROCenter(x%)
 1010 CLS:PRINT"Is message on disc? (Y/N
)"
 1020 IF FNYN="N" GOTO 1090
 1030 CLS:PRINT"Insert disc, then press
any key":G=GET
 1040 IF x%=2 GOTO 1070
 1050 *LOAD clear 4A38
 1060 ENDPROC
 1070 *LOAD crypt 4A38
 1080 ENDPROC
 1090 CLS:PRINT"Type message, ending wit
h |":V%-&4A37
 1100 REPEAT
 1110 V%=V%+1:G%=GET:IF G%=127 V%=V%-2 E
LSE ?V%=G%
 1120 PRINTCHR$(G%);
 1130 UNTIL G%=124
 1140 ENDPROC
 1150 :
 1160 DEF PROCoutput(x%)
 1170 A$(1)="encrypt":A$(2)="decrypt"
 1180 CLS:PRINT"Save "A$(x%)" to disc? (
Y/N)"
 1190 IF x%=2 PRINT"otherwise print"
 1200 IF FNYN="N" GOTO 1320
 1210 IF x%=2 N%=1:GOTO 1230
 1220 INPUT"How many copies? "N%
 1230 FOR M%=1 TO N%
 1240 PRINT"Insert disc, then press any
key":G%=GET
 1250 IF x%=1 F%=OPENOUT"crypt" ELSE F%=
OPENOUT"decrypt"
 1260 W%=&3A97
 1270 REPEAT
 1280 W%=W%+1
 1290 BPUT#F%,?W%
 1300 UNTIL ?W%=124
 1310 CLOSE#F%:NEXT M%
 1320 CLS:IF x%=2 GOTO 1340 ELSE PRINT"P
rint-out required? (Y/N)"
 1330 IF FNYN="N" ENDPROC
 1340 IF x%=2 N%=1:GOTO 1360
 1350 INPUT"How many copies "N%
 1360 VDU2:FOR M%=1 TO N%
 1370 W%=&3A97
 1380 REPEAT
 1390 W%=W%+1:PRINTCHR$(?W%);
 1400 UNTIL ?W%=124
 1410 CLS:PRINT"Paper - then press any k
ey":G=GET
 1420 NEXT M%
 1430 VDU3
 1440 ENDPROC
 1450 :
 1460 DEF PROCenciph
 1470 PROCenter(1)
 1480 U%=&5BCA:W%=&3A97:X%=0:V%=&4A37
 1490 REPEAT
 1500 V%=V%+1:G%-?V%
 1510 IF G%-124 GOTO 1630
 1520 IF G%<>32 GOTO 1550
 1530 REPEAT:V%=V%+1:G%=?V%:UNTIL G%<>32
:V%=V%-1:G%=74
 1540 IF G%=124 GOTO 1630
 1550 IF G%<65 OR G%>122 OR (G%>90 AND G
%<97) GOTO 1500
 1560 G%=G% AND 223
 1570 U%=U%+2
 1580 T%=256*(?U%)+?(U%+1)
 1590 H%=(G%+T%) MOD 26 +65
 1600 W%=W%+1:?W%=H%
```

continued on page 18

Weather Station (Part 2)

Nick Case keeps track of just how Flaming June is with his Beeb.

This program is an adaptation of last month's BAROM, so that you can keep a record of those outside air temperatures from a maximum and minimum thermometer, and add to your weather station records.



The real weather

As before you can use your Beeb to display the graph and jump from month to month or year to year, and then dump your choice to a printer, if required.

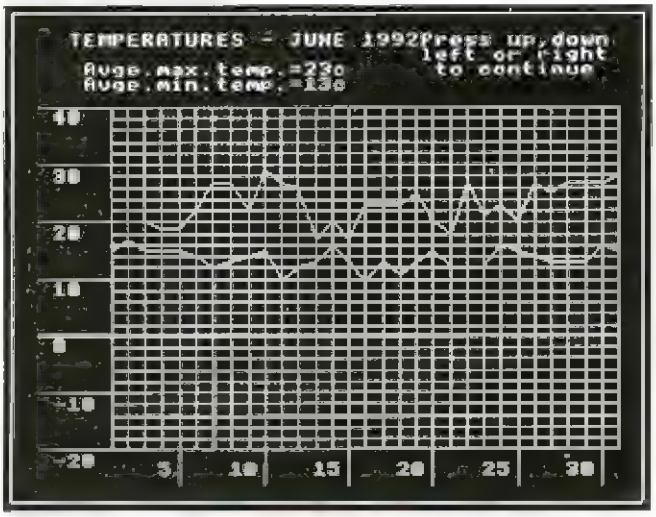
Load the BAROM program from last month, and assuming you kept to the same line numbers as given, just change the lines printed in the listing given this month.

Save the new program as MAXMIN. If it is all correct then when you run it you should get the display for my home town for January 1988. As before, just start noting down those thermometer readings every day and soon you will have your first month's worth. Edit them into the DATA lines of the program, save the new program as <month>'<year>, e.g. MAR'92, and away you go.

(N.B. Keep the BAROM and MAXMIN programs on seperate discs)

As before, if you want some quicker results, then look in back numbers of your local newspaper, etc.

The cursor keys control the displayed month, as before. Until you have built up the necessary data, to stop the display locking up, put a REM at the start of lines 190 to 220 as required.



Flaming June?

PROGRAM NOTES

The DATA for maximum and minimum temperatures are kept separate on lines 2420 and 2430, as PROCplot is now called twice, first for the maximum and then for the minimum. Also, please note that the total number of days in the month required must be doubled and put in line 2410 (e.g. for June put 60,for July put 62). Let's hope we have some temperatures worth recording!

```
   10 REM Program  MAXMIN
   40 REM BEEBUG   July 1992
 1010 DIM X(62),Y(62)
 1040 hght%=25:wdth%=37
 1110 COLOUR2:PRINTTAB(2,1)"TEMPERATURES
.
 1210 FOR y%=0 TO 30:MOVE 160,80+(y%*hgh
t%):DRAW 160+(30*wdth%),80+(y%*hght%):NE
```

Weather Station (2)

```
XT
 1220 FOR x%=0 TO 30:MOVE 160+(x%*wdth%)
,80:DRAW 160+(x%*wdth%),80+(30*hght%):NE
XT
 1310 FOR y%=0 TO 30 STEP5:MOVE 0,80+(y%
*hght%):DRAW 160,80+(y%*hght%):NEXT
 1330 MOVE 0,335:DRAW 1270,335
 1340 ENDPROC
 1350 :
 1420 FOR y%=0 TO 30 STEP5:MOVE 30,70+(y
%*hght%):PRINT;(y%*2)-20;:NEXT
 1430 ENDPROC
 1440 :
 1550 MOVE 160+((X(1)-1)*wdth%),330+(Y(1
)*(hght%/2)):FOR N=2 TO (end/2):GCOL0,1
 1570 MOVE 160+((X(1)-1)*wdth%),330+(Y(e
nd/2+1)*(hght%/2)):FOR N=(end/2+1) TO en
d:GCOL0,2:PROCplot(X(N),Y(N)):NEXT
 1580 ENDPROC
 1590 :
 1610 VDU5:PLOT5,160+((X(N)-1)*wdth%),33
0+(Y(N)*(hght%/2)):VDU4
 1620 ENDPROC
 1630 :
 1700 DEFPROCsum
 1710 totalmax%=0:FOR N=1 TO end/2:total
```

```
max%=totalmax%+Y(N):NEXT N
 1720 meanmax%=totalmax%/(end/2):PRINTTA
B(3,3)"Avge.max.temp.=";meanmax%;:PRINT"
C"
 1730 totalmin%=0:FOR N=(end/2+1) TO end
:totalmin%=totalmin%+Y(N):NEXT N
 1740 meanmin%=totalmin%/(end/2):PRINTTA
B(3,4)"Avge.min.temp.=";meanmin%;:PRINT"
C"
 1750 ENDPROC
 1760 :
 2340 REM Replace the following with
your own data.
 2350 :
 2400 DATA JAN,88
 2410 DATA 62
 2420 DATA 1,10,2,10,3,10,4,9,5,9,6,8,7,
7,8,7,9,12,10,9,11,7,12,6,13,7,14,10,15,
7,16,3,17,3,18,3,19,5,20,10,21,8,22,7,23
,11,24,13,25,10,26,7,27,5,28,7,29,7,30,7
,31,8
 2430 DATA1,5,2,7,3,5,4,5,5,5,6,4,7,2,8,
-3,9,10,10,5,11,0,12,2,13,3,14,6,15,3,16
,0,17,-3,18,1,19,3,20,5,21,-2,22,2,23,-2
,24,6,25,5,26,4,27,3,28,3,29,2,30,0,31,0
```

High Security Ciphers 1 (continued from page 16)

```
 1610 X%=X%+1
 1620 IF X%=5 H%=32:W%=W%+1:?W%=H%:X%=0
 1630 UNTIL G%=124
 1640 PROCoutput(1)
 1650 ENDPROC
 1660 :
 1670 DEF PROCdeciph
 1680 PROCenter(2)
 1690 U%=&5BCA:V%=&4A37:W%=&3A97
 1700 Z%=U%
 1710 PRINT"Press Shift to advance displ
ay.":PRINT:VDU14
 1720 REPEAT
 1730 V%=V%+1:G%=?V%
 1740 IF G%=124 GOTO 1870
 1750 IF G%<>32 GOTO 1790
 1760 U%=U%+10:Z%=U%
 1770 V%=V%+1:G%=?V%:IF G%=32 GOTO 1770
 1780 IF G%=124 GOTO 1870
```

```
 1790 Z%=Z%+2
 1800 T%=256*(?Z%)+?(Z%+1)
 1810 S%=130+T%-G%
 1820 IF S% MOD 26=0 H%=S% DIV 26 ELSE H
%=S% DIV 26 +1
 1830 H%=26*H%+G%-65-T%
 1840 IF H%=74 H%=32
 1850 PRINTCHR$(H%);
 1860 W%=W%+1:?W%=H%
 1870 UNTIL G%=124
 1880 PRINT:PRINT:PRINT"Press space bar
to stop here, or any "'"other key to con
tinue":G%=GET:IF G%=32 END
 1890 PROCoutput(2)
 1900 ENDPROC
 1910 :
 1920 DEF FNYN
 1930 K%=GET AND 223:IF K%<>78 AND K%<>8
9 THEN 1930 ELSE K$=CHR$(K%):=K$
```

Corkscrew: Procedure Manager

Ian Palmer builds a procedure library for Basic.

Writing Basic programs can be faster if you have, and use, a routine library, perhaps containing procedures from BEEBUG's own library. Another advantage of a library is that the programs tend to be better, for example a library text input routine that allows cursor editing takes little more effort to call than a simple INPUT command, but the quality of the program is improved dramatically.

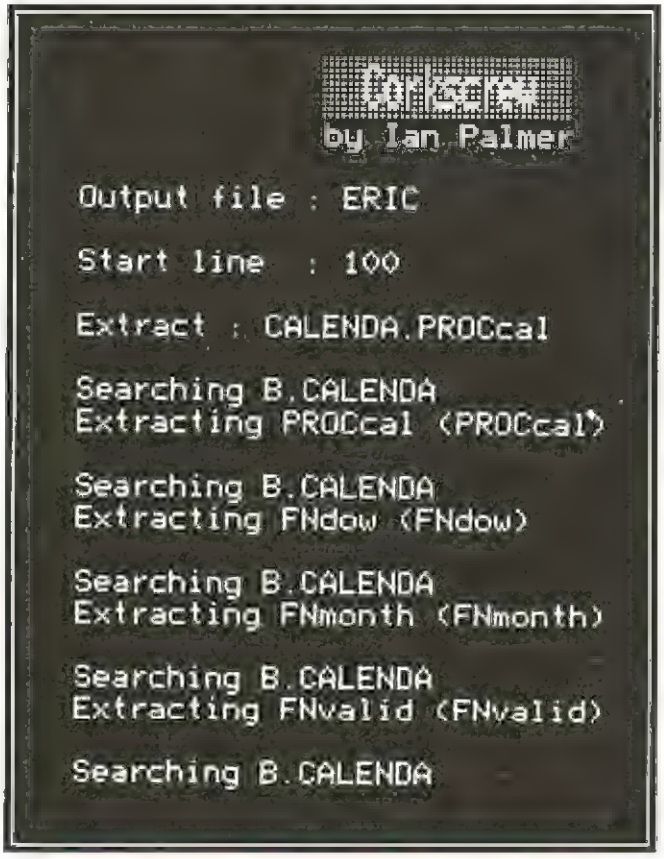
The major problems with routine libraries in BBC Basic are those of organization and retrieval. On the organization front you can either group together related routines into files, which leads to time consuming extraction of the routines you require. Alternatively you can place each routine in a file of its own, which leads to many files and problems in locating the routine you want from a barrage of file names (which can be no more than 7 characters long on a DFS system).

PROBLEMS, PROBLEMS

Whichever method of organization you use, there are more problems that are common to both. First there is the problem of line numbers; generally it is better to store the routines without line numbers, but this requires adding an AUTO line to the front of the file before using *EXEC to put it into your program. Another problem is remembering to extract lower level routines which the selected routine uses, and any further routines that these depend upon, and so on. The final problem is that of name clashes. If a routine you want to extract from your library has the same name as an existing routine in your program, you need to change the name of one of the procedures, and all the calls to it.

SO, WHY BOTHER?

It is easy to see why libraries, if created, are under-used - it is often faster to re-write the routine than extract it from a library - until now that is. Say hello to *Corkscrew*, the automated library routine extraction program.



Extracting the Calenda procedures

Before you can use Corkscrew, however, you need to construct your library of routines in such a way that Corkscrew can cope with them. Related routines are grouped together in an ASCII file with each line of Basic on a separate line but without line numbers. - for example, you may have a file called 'IO', another called 'MATHS', and so on. Each routine (procedure or function) in a library file must contain three parts: a header, a description, and a body.

A header informs Corkscrew of the name of this routine, and the names (and files in which they are to be located) of all further routines that this routine calls. Each routine is given a number (0 to 25), where zero must be the routine in

question, and numbers 1 to 25 are assigned to any called routines you like. As an example, take the following routines which are to be added to the library 'MATHS'.

```
DEFFNmean(N%)
LOCAL T%
T%=FNsum(N%)
=T%/N%
:
DEFFNsum(N%)
LOCAL T%,A%:T%=0
FOR A%=1 TO N%:T%=T%+FNgetitem(A%)
NEXT:=T%
:
```

In this example suppose that FNgetitem cannot be placed in a library as it is specific to the program to which these library routines are to be added. The header for FNmean would be as follows:

```
#0 FNmean
#1 MATHS.FNsum
```

As can be seen the definitions of the called procedures specify the library file in which the routine is to be found. This means that routines can call other library routines that are not in the same file as the caller. The header for FNsum would be:

```
#0 FNsum
#1 @.FNgetitem
```

Note that FNgetitem cannot be located in any library file, so the null file '@' is used to specify that it will be supplied fresh in the program being constructed.

Next comes the description which can be anything you like but the first line must start with '#!'. The description is designed to remind the user of the operation performed by the routine, and the parameters that should be supplied. A suggested format for a description is shown by the following example for FNmean:

```
#! ( number )
Returns the mean value of a given
number of items.
```

Next comes the body, which is the code of the routine. Every occurrence of a routine call must be replaced by a hash followed by the number of that routine from the header. Thus the body of FNmean becomes:

```
#<
DEF#0(N%)
LOCAL T%
T%=#1(N%)
=T%/N%
:
#>
```

The '#<' and '#>' mark the start and end of the body respectively, and must be placed on lines of their own. If the code contains a hash followed by a number, that occurs outside quotes, then two hashes must be placed before the number, so:

`CLOSE#0`	becomes	`CLOSE##0`
`CLOSE#C%`	stays as	`CLOSE#C%`
`"item #1"`	stays as	`"item #1"`

You should have noticed by now that routines are stored without line numbers. If you already have library routines that have line numbers, then these line numbers need to be removed. If you have Acorn's Edit then this is easy to do. Simply load all the routines you wish to place in a file into Edit. Then select 'Global Replace' and type the following at the prompt:

```
$^[ 0-9]/$
```

This will remove any line numbers from all lines other than the first. Notice the space before the zero. Then you can add headers and descriptions for each routine, and you have your first library file. The ASCII library files should be stored in a directory called *BLIB* if you have ADFS, or simply *B* if you have DFS.

Two programs accompany this article, the first is called List, the second is Corkscrew. List allows you to produce hard copies of the contents of each of your library files. When you run it, it asks you for the name of a library. Type it in (checking first that your printer is

switched on) and the program will do the rest.

Corkscrew, on the other hand, actually performs routine extraction. When you run it, it first asks you for the name of a file to use for output. This should be anything other than *Text* as the program produces a file with this name which contains the titles and descriptions of all routines extracted.

You are then asked for a line number. This defines the line number of the first line of the output and all subsequent lines rise in increments of ten.

Next you are given the prompt *Extract :*. This is asking you for the first routine to extract from your library. The format of your input should be one of the following:

```
FILE.ROUTINE
FILE.ROUTINE (ROUTINE)
FILE.ROUTINE (FILE.ROUTINE)
```

For example if you want to extract FNmean, then simply type 'MATHS.FNmean' at the prompt. Corkscrew will then search the file *MATHS* for the routine and place it in the output file. It will then automatically extract FNsum, as this is required by FNmean.

As another example, suppose you wish to extract FNmean and you already have a function FNsum but you still need to extract FNsum from the MATHS library as *your* FNsum does not perform the right function. First you need to inform Corkscrew that you have a routine called FNsum. This is done by typing '@.FNsum' at the *Extract* prompt. Next you type 'MATHS.FNmean' as normal, and Corkscrew will extract this routine, but when it comes to extract MATHS.FNsum it will note that it needs to change its name and it will do so (by adding a '1'), and as if by magic the name will also be changed in the call to it in FNmean.

```
100DEFPROCcal(M%,Y%)
110LOCAL D%,S%,X%:X%=POS
120S%=FNdow(1,M%,Y%):IF S%=0 ENDPROC
130PRINTSPC(6);FNmonth(M%);" ";Y%''TAB
(X%,VPOS);"Su Mo Tu We Th Fr Sa"''TAB(X%
+(S%-1)*3,VPOS);:FOR D%=1 TO 31
140IF FNvalid(D%,M%,Y%) PRINTFNfill(D%
,2);" ";:S%=S%+1:IF S%=8 S%=1:PRINT'TAB(
X%,VPOS);
150NEXT:ENDPROC
160:
170DEFFNdow(D%,M%,Y%):LOCALQ%,N%,X%:N%
=M%:X%=Y%:IFM%<=2 N%=M%+12:X%=Y%-1
180Q%=(D%+2.6*(N%+1)+X%+(X% DIV 4)-(X%
DIV 100)+(X% DIV 400)) MOD 7
190IF Y%<1752 OR (Y%=1752 AND (M%<9 OR
(M%=9 AND D%<3))) Q%=Q%-3
200IF Q%<1 Q%=Q%+7
210IF NOT(FNvalid(D%,M%,Y%)) Q%=0
220=Q%
230:
240DEFFNmonth(N%)
250IF N%<1 OR N%>12 ="***"
260=MID$("JanFebMarAprMayJunJulAugSep0
```

The Calenda procedures in your program

There is another way to achieve the above, which allows you (rather than Corkscrew) to select what name to change MATHS.FNsum to. This involves typing 'MATHS.FNsum (FNwhatever)' at the prompt, where 'FNwhatever' is the name you wish to use. FNsum will then be extracted from MATHS, and its name will be changed. Next you need to extract FNmean, so simply type 'MATHS.FNmean' at the prompt and this routine will be extracted. Again as if by magic, the call in FNmean to FNsum will be changed to whatever you've renamed it.

The next example involves a case where your version of FNgetitem is called FNitem. This is handled by first typing:

```
@.FNgetitem (FNitem)
```

at the prompt. No action will be taken by Corkscrew, but the name change will be noted. Next extract FNmean as normal.

The final example involves the case where although FNsum expects FNgetitem to be written especially for the new program, it (in this case) actually

resides in a library file (say *DATABASE*) and is called FNreaditem. This too can be handled by Corkscrew, by first typing:

```
@.FNgetitem (DATABASE.FNreaditem)
```

at the prompt. Corkscrew will then extract FNreaditem from *DATABASE*. Next you need to extract FNmean as normal, and the relevant name change will be made.

Fairly complex substitutions can be performed by Corkscrew, although the order in which you type in the extraction commands is very important. For example:

```
MATHS.FNsum (@.FNsum)
@.FNsum (DATABASE.FNtotal)
```

is not the same as:

```
@.FNsum (DATABASE.FNtotal)
MATHS.FNsum (@.FNsum)
```

In the latter case MATHS.FNsum will be replaced by DATABASE.FNtotal, but in the former case MATHS.FNsum is simply replaced by @.FNsum. This is because each substitution only inherits the effects of substitutions defined before it. In this way it is impossible to get a cyclic substitution, that is to say:

```
@.FNone (@.FNtwo)
@.FNtwo (@.FNone)
```

will not cause any problems.

Once you have finished extracting all the routines you require, simply press Return at the prompt and Corkscrew will then close all files and exit.

It is worth noting that Corkscrew remembers all the routines it has extracted during a session, and will not extract again a routine it has already found. Thus it is best to extract all the required routines for a program in one session, as this will avoid the possibility of the same routine being extracted twice, and will also help to avoid name clashes.

Also included on this month's disc are two library files called *CALENDA* and *MEMORY* which must be placed in the B or BLIB directory. *CALENDA* has various routines related to a calendar. For example PROCcal will display the calendar for any month-year combination you give it. All the routines expect years to be given in full (for example 1991, not 91) and months to be given as a number between 1 and 12. The routines assume the current rules for leap years for all years, and so calendars may be incorrect for years centuries ago, although they do make corrections for the transference between Julian and Gregorian calendars. To see the effect of this and get some idea of how the whole thing works try this.

Run Corkscrew and open an output file. Give a sensible start line and then, at the *Extract* prompt type CALENDA.PROCcal. PROCcal and all necessary routines will be extracted. At the next *Extract* prompt press Return, type NEW then spool in your output file. Now you can call the procedure using, for example, PROCcal(9,1752).

Other routines in this library perform tasks such as returning the number of days between two given dates, returning whether or not a year is a leap year, finding the day of the week that a certain date fell upon, and so on. Full descriptions are included in the library.

MEMORY has routines to handle dynamic memory allocation and de-allocation within Basic programs. This gives greater freedom over the limited memory of the BBC computers.

After a little practice Corkscrew will help you set up new programs quickly and easily by stopping you having to re-invent the wheel each time. You will also get the chance to develop and refine your routines and include the upgrades in all your subsequent programs. Soon you'll wonder how you did without it!

Listing 1

```
  10 REM Program LIST
  20 REM Version B 1.0
  30 REM Author  Ian Palmer
  40 REM BEEBUG  July 1992
  50 REM Program subject to copyright
  60 :
 100 D$="BLIB.":REM For DFS D$="B."
 110 MODE128
 120 REPEAT
 130 INPUT'"Library : "I$
 140 IF I$<>"" PROClist(I$)
 150 UNTIL I$=""
 160 END
 170 :
1000 DEF PROClist(I$)
1010 LOCAL F$,A%,F%,Z%,i%:Z%=FALSE
1020 F$=D$+I$:i%=OPENIN F$
1030 IF i%=0 PRINT'"Can't open ";I$
1040 VDU2:PRINT"Library : ";I$'
1050 REPEAT:L$=FNinput(i%,Z%)
1060 IF LEFT$(L$,2)="#0" PRINT'RIGHT$(L
$,LEN(L$)-2);
1070 IF L$="#<" Z%=FALSE:PRINT
1080 IF Z% PRINTTAB(15);L$
1090 IF LEFT$(L$,2)="#!" Z%=TRUE:PRINTT
AB(14);RIGHT$(L$,LEN(L$)-2)
1100 UNTIL EOF#i%
1110 CLOSE#i%:VDU3
1120 ENDPROC
1130 :
1140 DEF FNinput(c%,W%)
1150 LOCAL Q$,I%:Q$="":IF EOF#c% THEN =
Q$
1160 REPEAT:I%=BGET#c%:IF I%<>13 AND (I
%<>32 OR W%) Q$=Q$+CHR$(I%):IF I%=33 AND
 W%=FALSE W%=TRUE
1170 UNTIL I%=13 OR EOF#c%
1180 =Q$
```

Listing 2

```
  10 REM Program Corkscrew
  20 REM Version B 1.0
  30 REM Author  Ian Palmer
  40 REM BEEBUG  July 1992
  50 REM Program subject to copyright
  60 :
 100 D$="B.":REM BLIB. FOR ADFS
 110 MODE135:DIM I$(25),C$(25)
 120 B%=0:P%=0:T%=0
 130 PROCtitle(CHR$(132)+CHR$(157)+CHR$
(135)+"Corkscrew  "+CHR$(156),1):PROCcen
tre("by Ian Palmer",3):VDU28,0,24,39,5
 140 INPUT"Output file : "F$:o%=OPENOUT
 F$:INPUT'"Start line  : "L%:t%=OPENOUT"
TEXT"
 150 ON ERROR IF ERR=17 GOTO160:ELSE MO
DE7:PRINT:REPORT:PRINT" at line ";ERL:EN
D
 160 REPEAT
 170 INPUT'"Extract : "I$
 180 IF I$<>"" IF FNfindcall(I$)="" C$=
FNinsert(I$):P%=FNnext(P%):PROCimport($(
P%+2),C$)
 190 IF FNnext(P%)<>0 REPEAT:P%=FNnext(
P%):J$=$(P%+2):C$=$(P%+3+LEN(J$)):PROCim
port(J$,C$):UNTIL FNnext(P%)=0
 200 UNTIL I$="":CLOSE#o%:CLOSE#t%
 210 END
 220 :
1000 DEF FNnext(P%):IF P%=0 =B%
1010 =256*?(P%+1)+?P%
1020 :
1030 DEF FNinsert(I$)
1040 LOCAL C$,T$,R$,Q%
1050 R$="":IF INSTR(I$,"(") R$=MID$(I$,
INSTR(I$,"(")+1,INSTR(I$,")")-INSTR(I$,"
(")-1):I$=LEFT$(I$,INSTR(I$," ")-1)
1060 IF R$="" R$=FNfindcall(I$):IF R$<>
"" AND INSTR(R$,".")=0 =R$
1070 IF R$="" C$=FNcheck(RIGHT$(I$,LEN(
I$)-INSTR(I$,".")+1)):ELSE C$=R$:IF INSTR(
R$,".") R$=FNinsert(R$)
1080 DIM Q% (LEN(I$)+LEN(C$)+4):IF T%<>
0 ?T%=Q%:?(T%+1)=Q% DIV 256:ELSE B%=Q%
1090 T%=Q%:$(T%+2)=I$:$(T%+LEN(I$)+3)=C
$:?T%=0:?(T%+1)=0
1100 =C$
1110 :
1120 DEF FNcheck(C$):LOCAL A%,F%,T$
1130 REPEAT:F%=FALSE
1140 A%=B%:IF T%>0 REPEAT:T$=$(LEN($(A%
+2))+3+A%):A%=FNnext(A%):UNTIL T$=C$ OR
A%=0:IF T$=C$ F%=TRUE
1150 IF F% C$=C$+"1"
1160 UNTIL NOT(F%)
1170 =C$
1180 :
```

Continued on page 56

BEEBUG Education

by Mark Sealey

INTRODUCTION

As outlined in BEEBUG Education (Vol.11 No. 1), there are still some producers of educational software for the 8-bit series of machines. We reviewed then two items from the Scottish Council for Educational Technology (SCET). This month we take a look at two more, *Style* and *Sixth Sense*.

Products	**Style and Sixth Sense**
Supplier	**SCET, Scottish Council for Educational Technology, 74 Victoria Crescent Road, Glasgow G12 9JN. Tel. 041-334 9314**
Price	**Style from £35.00 ex. VAT Sixth Sense £35.00 ex. VAT**

Style

Style - believe it or not - is a word processor which exists both as a ROM and on DFS (40 or 80 track) discs. If you compare the visual feel of Style with other such packages already available for the BBC and Master, your first reaction is how much it owes to developments that followed the release of View and Wordwise (Plus): it operates through pull-down menus, is quite rich in on-screen information, has an option to send text in Electronic Mail format (i.e. it weeds out accents and other characters with special meanings that could confuse comms software) and, significantly, will allow accurate representation of underline, bold, italic effects, etc. on screen.

FULL FEATURED WYSIWYG

Style is a fully WYSIWYG word processor, and is all the better for it. Once booted, you work in a black on cyan screen with a simple and attractive, uncluttered environment. All the usual operations like insert/overtype, block (including copy, move and save) and ruler manipulation, variable margins and tabs, and the four types of justification are available. Cursor control is standard using combinations of Shift and Ctrl with the cursor keys. Text is formatted automatically, and wordwrap is applied by default.

A somewhat limited hanging indent facility exists, which together with decimal tab stops and local centring make the creation of tables much easier than with Wordwise.

There is a global and selective search-and-replace, and again all styles are available in headers and footers. Printing offers mailmerge (from a file only - created in Style itself). NLQ (or your printer's equivalent) and similar can be selected, as can other drivers, from the software. OS (*) commands are also possible so you can catalogue a disc in order to load a file without leaving the program. However, an essential which seems to be missing is the facility to select a file to load from a menu of those available.

WORKING WITH STYLE

Most operations are obtained by the function keys, and movement around the text is easy and intuitive. It is especially convincing to be able to see all your text style effects instantly visible as you type: there is an indication on the ruler line as to which style (italic, underline, bold or a combination etc.) is currently selected, and of the space left for text.

There is a utilities disc for running, among other things, the mailmerge, all of

the operations of which have a simple but sophisticated feel to them.

The discs are unprotected, and will even transfer successfully to run under the BBC Emulator (!65host) on the Archimedes series machines. This is more of a bonus than it at first sounds because many schools now use more than one system and will be pleased to be able to transfer data and skills from one to the other.

APPLICATIONS

Included with Style there are three A4 manuals (getting started, reference and a training volume, the latter full of detailed worked examples from the world of modern language and european studies). There is a full work disc to support each of four such projects centred around EEC activities.

The documentation (as that of the package as a whole) is of a commendably high standard. If you are still looking for a word processor that is easy and intuitive to use and will satisfy both simple and more sophisticated requirements, then - even after all this time - Style might just fit the bill.

Sixth Sense

To quote from the excellent manual that accompanies this sensing package, also from SCET, "You know the problem - the classroom is cold on a winter's morning. By lunchtime, it is getting warmer and by three o'clock, it seems at its hottest.... If you strapped in those... sensors... to the machine and ran some sort of program to capture... data for a day and then printed it out a graph, you might be able to find out all the details..., the details that you needed to make the case for air conditioning in your west facing classroom".

Sixth Sense is, in fact, just the package to do this. It is a data capture and display package with both specific applications for science, geography and technology involving the immediate environment, and applications to support general understanding of the ways in which the microcomputer can be used to interface with the real world.

THE SOFTWARE

The manual claims that the software can be used with most of the analogue port interface boxes currently available (e.g. Deltronics and Harris). It uses mode 4, and produces a graph in any of four types (line, points, line-points, and bar), plus the facility for overlaying two graphs. It can zoom in on a part of the current graph and produce a table of the data which has generated it.

SUPPORT MATERIAL

The publishers acknowledge that data capture and analysis may well be a new (and threatening) area for many teachers. The ideal situation would be if some form of examples existed, leading the experimenter gently through the procedure and then suggesting the learning and teaching points, the questions to ask and finally the essence of the scientific conclusion.

SCET has produced just this in a set of sample files mirroring typical activities for middle and secondary school children, although older children in those primaries rich enough to own interface boxes could also cope.

For instance the history national curriculum requires children to work on

Continued on page 40

LOGO and the Family Tree

Geoffrey Sherman discovers he's 128th in line for the throne with the help of LOGO.

I recently became interested in genealogy and for recording details of my family's history I needed a program to store data. I considered using my existing database and word processor software as Mr H. Goodman described in BEEBUG Vol.8 No.2, but although this arrangement works, I felt that it might be more interesting to have a program dedicated to the purpose. Before searching for published genealogy software, I decided to try writing a suitable program myself.

Tree main menu

To many people, at least outside the education world, the word LOGO is synonymous with 'Turtle Graphics' and no more. Having purchased a copy of Acornsoft's LOGO language some time ago, I soon realised that it had quite a comprehensive facility for list processing and in fact the language is really based around words and lists. Since family tree data and individual's personal details are really only a collection of words and linked lists, LOGO seemed an ideal language for this program. As an introduction to the language I therefore set about writing a family tree program using it.

LOGO LANGUAGE

To my surprise LOGO was easy to learn, user friendly and a much better language to handle word lists than Basic. For those unfamiliar with the language it is based around procedures, as in Basic, but without line numbers. Programs therefore have to be well structured. The beauty of this approach is that each procedure is tested individually to check it works as required; there is no need to run the whole program to do this. LOGO has a built-in editor to edit procedures, and best of all, view and edit variables too. Since variables are often lists, these can be inspected to check how they are being created or changed by the program.

The menu for one person

Acornsoft's LOGO provides a useful facility which allows one to load procedures and data directly from disc into an existing program as an *overlay*, without erasing a program already in memory. In the same way one can delete other procedures and variables from a program to make space for new ones quite easily. This is much friendlier than the method needed in Basic (see the *Corkscrew* article elsewhere in this issue) and makes the manipulation of memory space that much simpler. It is also possible to save variables to a separate

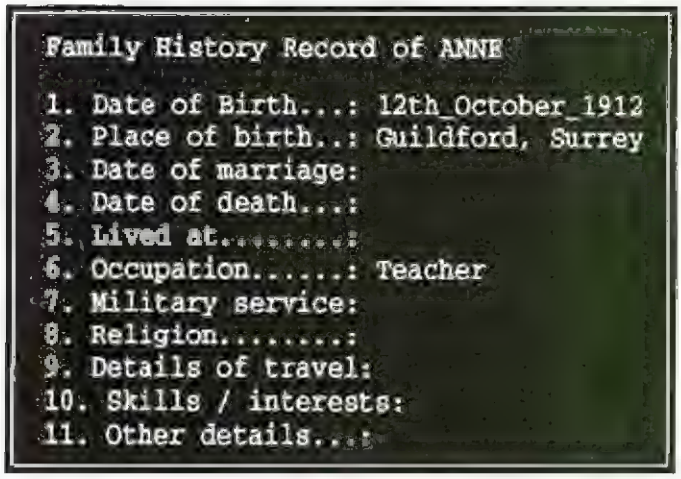
file. This made the manipulation of data files for different generations straightforward.

FAMILY TREE PROGRAM

Before embarking on the program, I had to decide what I wanted the program to do. My aims were to:

1. Store names of family members as a GENERATION.
2. Store details of the FAMILIES of each member of a generation.
3. Store PERSONAL DETAILS about each individual.
4. Plot the FAMILY TREE.

Generation 5



Anne's individual details

As already discussed, LOGO handles words and lists. The next step was to decide on the list structure. As this was my first encounter with LOGO I decided to keep things simple. For those unfamiliar with lists here is a short explanation.

A list is defined as a sequence of elements separated by spaces. Each element can be either a word or another list. Thus a list can be a collection of words, words and sublists or just sublists. The list is contained within square brackets (i.e. []). Here is an example:

```
"EXAMPLE is  [THIS IS [A LIST]]
```

The initial " tells LOGO that this is a variable. The name of the list is "EXAMPLE and it has 3 elements i.e. THIS, IS and [A LIST]. The last element is also a list which has two elements.

I decided to have three list types for the data lists to coincide with items 1, 2, and 3 above. These would be arranged as follows.

The first type would be the generation list "GENx where x is the generation number. This list would be of the form:

```
"GENx is [Anne Beryl Colin Derek]
```

where Anne, Beryl, Colin, and Derek are the names of each member of the family for generation x, i.e. the sons and daughters.

The second list type would be the family lists. These would be of the form:

```
"Anne is [Anne William [Paul Mary
James]]
```

where Anne is the family member for this generation, William is her spouse and Paul, Mary and James are their

offspring. Note that a *sublist* (i.e. an element of a list that is itself a list) is used to hold the offspring for reasons given later.

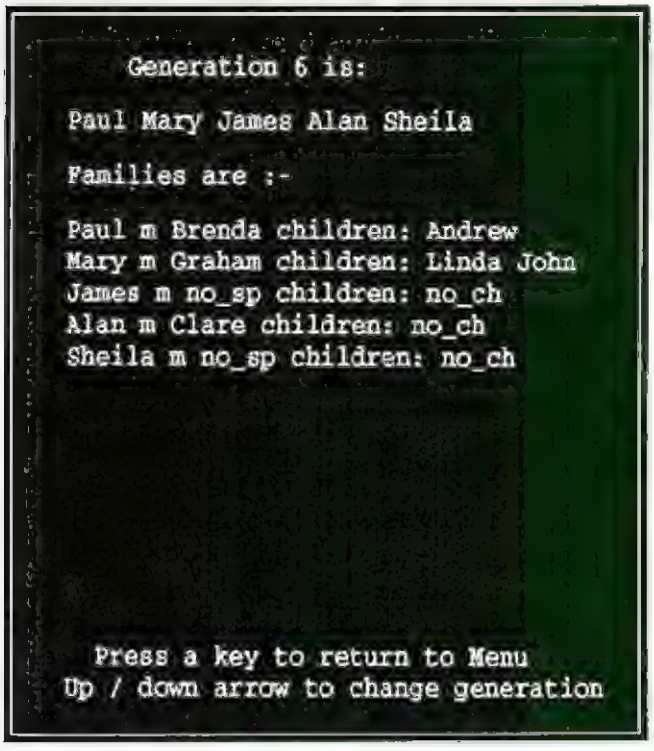
The third list type is the personal detail list. This list is given the same name as that for the member but a P is added in front. The list is first set up as a list of 12 empty lists thus:

```
"PAnne is [[] [] [] [] [] [] [] [] [] [] [] []]
```

An empty list is shown as []. I have chosen to record information in 12 fields as for a database with field names such as Date of Birth, Place of Birth, Occupation, etc. Each of the sublists in the personal detail list corresponds to one field hence "PAnne might look like this:

```
"PAnne is [[12th_October_1912] [Guildford,Surrey] [Teacher] [] [] [] [] [] [] [] [] []]
```

with the first 3 fields completed. Of course one may not be able to complete all fields as data is often hard to come by!



Generation 6

Note that no spaces can be entered between a group of words if these are to be regarded together as one element; the inclusion of spaces tells LOGO these are separate elements. I used the underline character '_' instead to join words.

The program is initiated by requesting the names of all members of a given generation as in GENx above. The program then extracts each member in turn and asks for the names of their spouse and then the names of their offspring. The first family list is then created automatically.

Changing things

The program continues by extracting the next family member from GENx and repeats the request for spouse and offspring names. In this way all the family lists for generation GENx are created.

The program then seeks approval to create the next (younger) generation. If confirmed this is performed using the family lists just created for generation GENx. The next generation, GENx+1, is created by extracting the last element (sublist) of the family list containing the names of the offspring of each family. It then combines these to form the new GENx+1 list. A simple example should clarify this.

```
"GEN5 is [Anne Beryl]
"Anne is [Anne William [Paul Mary
James]]
"Beryl is [Beryl Jim [Alan Sheila]]
```

hence

```
"GEN6 is [Paul Mary James Alan Sheila]
```

Surnames have been omitted but can be added after the family member's name if desired, separated from the christian name by a '_'.

One menu option is to view or amend individual's personal details. A routine displays the field names, which are held in another list, and prints out those fields from the "PName list for which there are entries (see the "PAnne example above). An input routine then allows additions or changes to be made, these being inserted into the "PName list within the appropriate sublist.

The last option is to view the family tree diagram. This routine is not yet complete. The algorithm to search progressively through each branch of the tree, plot the tree and print names as it goes, is proving to be quite a brain teaser!

LOGO ON THE BBC

LOGO is available to run on both BBC micro and Master. I have found that in use it seems to use up memory quite quickly. Shadow memory is essential if you want to use modes 0 or 3 (shadow modes 129 or 131) and have a modest sized program. I have been using LOGO with a 6502 second processor but even so have only a maximum of 19.5K available on start up. My family tree program occupied 17K of memory at first, not leaving much space for data! I then appreciated the usefulness of the overlay techniques available and managed to cut the main program down to 7.5K and load routines from disc as necessary. Even so one can soon run out of data space even when loading just a few generations at a time, especially as the tree grows in size! As with any family tree diagram, one has to decide how many branches to follow for the tree not to become too unwieldy.

```
Family History Record of ANNE

1. Date of Birth...: 12th_October_1912
2. Place of birth..: Guildford, Surrey
3. Date of marriage: 15th May 1937
4. Date of death...: 3rd June 1966
5. Lived at.......: Church La,Godalming
6. Occupation.....: Teacher
7. Military service:
8. Religion.......: CoE
9. Details of travel:
10. Skills / interests: Gifted artist
11. Other details...: Had many paintings
 displayed at Guildford Arts Festival
```

Anne's individual details (complete)

ON THE DISC

As well as the Acornsoft LOGO program files for this article this month's disc contains an ASCII listing of the complete program called *Lprog*. You will also find an ASCII file of program notes to help you run it entitled *Lnotes*. Users of Logotron LOGO should be able to adapt the listing quite easily.

CONCLUSION

I have found my introduction to LOGO to be most enlightening, and that this is quite a versatile language. It has its limitations, especially with memory when modest size programs are developed. I hope this article has been of some interest to others who may not have considered LOGO for applications other than Turtle Graphics. For those who have this ROM and have not delved into it I would heartily recommend a try. It makes a pleasant change from Basic.

Public Domain Software

This month Alan Blundell's regular look at PD software covers applications software.

Last month, I promised to look at applications software such as word processors, databases and spreadsheets. There isn't a vast range of such software in the public domain, but there are several gems which I think are worth seeking out. For Master users, a word processor or spreadsheet would have to be pretty good to compete for attention over the built-in View and ViewSheet, but there are some items which will be of interest to these users too.

EASIBASE is a database system which first appeared in an early issue of *Fast Access*, a subscription-only, disc based magazine for BBC and Master users. It is a general purpose, random access database and makes use of pull-down menus and icons, making it fairly easy to use. Bill Woodall found the program sufficiently useful to spend the time necessary to add a number of improvements to the system, based on his practical experience of using it, to make it even more useful. Then he submitted it to BBC PD for publication. A bit of correspondence with Phoenix Software, the publishers of *Fast Access*, resulted in their permission being given for the program to be distributed by BBC PD, although they retain the copyright in the package.

Version 4 of the package allows for up to 29 fields of up to 3 lines each (about 100 characters) and an unlimited number of records (subject to disc space). Calculations can be performed on numeric fields, fast searches can be performed on any (or all!) fields and there is a built-in '*' command environment with a few commands specific to the package. Whilst the system does have its limitations (especially in comparison to the powerful, relational database management systems available on PCs, but also in comparison with commercial BBC packages like ViewStore or BEEBUG's own Masterfile), it probably represents the most comprehensive currently available as (quasi) PD. The whole package occupies over 80K of disc space, and comes with over 30K of documentation in text files.

Other database programs are available, for example Chris Pinnock's *Einstein* database, which has gone through several stages of improvement and is currently up to version 2.4. It supports both DFS and ADFS filing systems, can be used from a hard disc and makes use of a screen windows system built in to the program. A.S.Shakoor's *BBC DBase* is a simple card index program, which is handy for storing names and addresses (for example), but lacks search facilities.

Finally on the database front, the now extinct *Disk User* magazine published a database system called *Tracer*. I mention this because *Disk User* is/has been available from a couple of PD libraries who have been given permission to distribute by the original publishers, Argus Specialist Press, and other issues of the magazine contained ready-built databases for use with the program, such as the Periodic Table (Chemistry, for those whose school days are long forgotten!), British birds, bulletin boards, kings & queens of Britain and the Dewey

library classification system. The passage of time may have rendered much of the bulletin boards database obsolete, but the other examples mentioned are quite 'future-proof'.

Word processors and spreadsheets are a bit thinner on the ground, but Hector C. Parr has produced simple examples of both, together with a simple database which he included in a disc also containing a collection of mathematical programs and demonstrations of data handling written for use in connection with A-level Computing. There is also a multi-lingual word processing system by Adam Sandman, called *Ultraword+*, which is unfortunately for the Master Compact only. Adam has also produced a thesaurus system, *Wordfinder*, which it is probably fair to say is only really the beginning of a usable thesaurus: to be genuinely useful, it would benefit from some programming improvements, plus word and word association files of a much more comprehensive nature than those currently included. The basis of a system is there, though, if someone cared to continue and improve the project.

Andrew Pepperell, mentioned before in this column, has recently produced a 'dictionary access' program which has a usable dictionary (about 500K!) included. The programs used to access this dictionary are fast and efficient, and are designed to let you find matches for partial words (indicating a possible use if you like crosswords) plus other manipulations of the dictionary. Unfortunately, the dictionary can't be used directly in relation to your word processing, except by exiting your word processor and loading the dictionary access program.

However, there are a few spelling checkers available as PD - three to my knowledge, one written by Allan Kelly (*Watcher*, previously distributed as Telesoftware), *ViewCheck* by Wayne Clarke and the originally-named *SpellCheck*, initially written by me but since enhanced by others, including Eric L. deBourcier and Martin Burchell. The combined efforts have resulted in a quite fast and efficient checker which can work with View, Wordwise Plus and Wordwise Plus II files as well as plain ASCII text.

Finally, to round off your word processing, a custom-designed letterhead can give a better effect to your printed words. Piers Wilson was an early contributor to the PD scene for Acorn micros with his letterhead designer, written over 2 years ago now. He later produced a much improved version which streamlined the process of creating letterheads and which has been praised in comments which I have received. His program is another which has benefited from pooled efforts, as a version modified for better suitability on Master series machines has recently been sent to me by John Barker, and this is distributed with Piers' agreement.

I'm sure that more 'serious' applications software will appear in the public domain in the future; as with other areas which have already been covered in this column, new software is appearing all the time, changing the overall picture. When something new and interesting appears, you will hear about it. Next month, I will look at a mixed bag of software which I and others have found of particular interest. In the meantime, I would like to thank everyone who has written to me commenting about this series of articles for their good wishes.

Best of BEEBUG

Applications I Disc

Business Graphics - for producing graphs, charts and diagrams
Video Cataloguer - catalogue and print labels for your video cassettes
Phone Book - an on-screen telephone book which can be easily edited and updated
Personalised Letter-Headings - design a stylish logo for your letter heads
Appointments Diary - a computerised appointments diary
Mapping the British Isles - draw a map of the British Isles at any size
Selective Breeding - a superb graphical display of selective breeding of insects
The Earth from Space - draw a picture of the Earth as seen from any point in space
Personalised Address Book - on-screen address and phone book
Page Designer - a page-making package for Epson compatible printers
World by Night and Day - a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

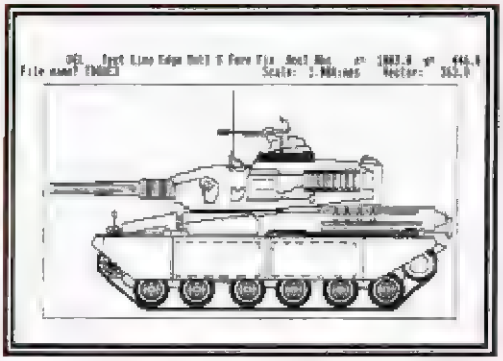
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.



ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

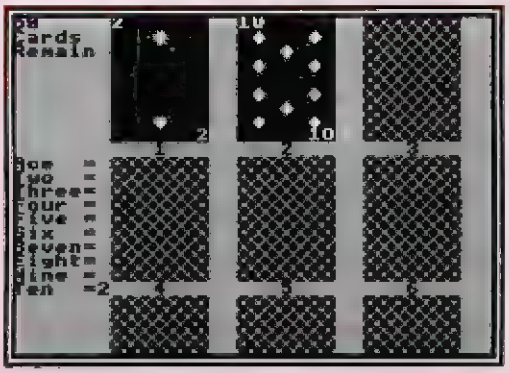
- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate. For p&p see Membership page.

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Board Games



Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

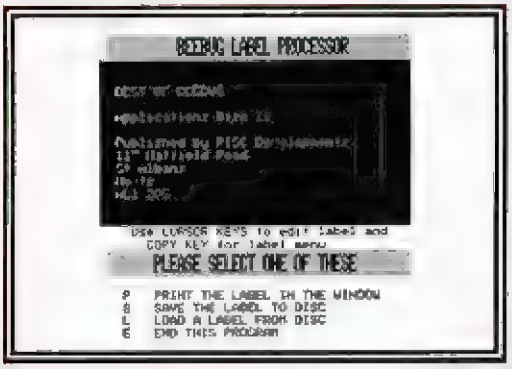
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Applications II Disc



Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes - generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures - calibrates and plots up to four temperatures

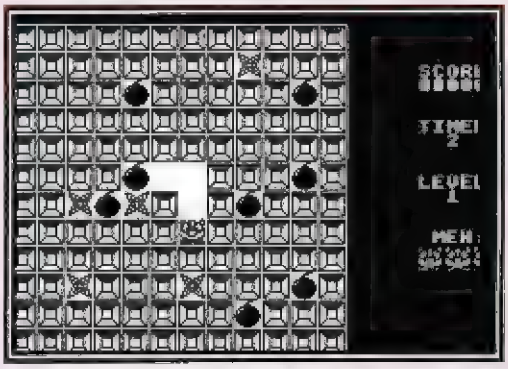
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

Label Processor - for designing and printing labels on Epson compatible printers

Arcade Games



George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

Best of BEEBUG

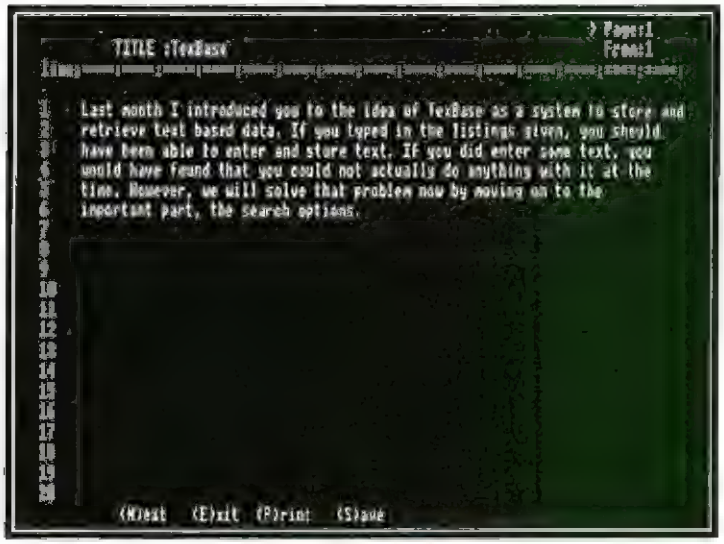
	Stock Code	Price		Stock Code	Price
File Handling for All Book	BK02b	£ 9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75	**File Handling for All** Disc (3.5" ADFS)	BK07a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£ 11.95**	**Joint Offer book and disc** (3.5" ADFS)	BK06b	**£ 11.95**
Magscan (40 DFS)	0005a	£ 9.95	**Magscan Upgrade** (40 DFS)	0011a	£ 4.75
Magscan (80T DFS)	0006a	£ 9.95	**Magscan Upgrade** (80T DFS)	0010a	£ 4.75
Magscan (3.5" ADFS)	0007a	£ 9.95	**Magscan Upgrade** (3.5" ADFS)	1458a	£ 4.75

All prices include VAT where appropriate. For p&p see Membership page.

Enhanced TexBase Search Routine

Howard Javes gets TexBase to widen its search.

The TexBase database, published in BEEBUG, is very useful for storing and retrieving text. I wanted to use the program to store details of my album collection, including the names of each track on the disc or tape. I then wanted to be able to search for a particular track and find out which album it was on. With the usual search program this would not be possible as it would require almost every word to be entered as a keyword. To overcome this problem I have written some enhancements to add to the original search program in order to allow me to search for a word in the text itself or in the title. It is still possible to search in keywords as before, with or without searching in text.



The Search option

ENTERING THE ENHANCEMENTS

The enhancements are in two listings - a short assembly language program and a set of lines to be added to the original program from BEEBUG Vol. 10 No. 5. I will assume that the original TexBase programs have already been typed in and saved. First, type in and save the assembler listing, *SOURCE*. This need not be on the same disc as the TexBase package. Now put the TexBase disc in the drive and run the *SOURCE* program. This will save a machine code program which is used by the main program. Now type in the *AMEND* listing and save it. When you are sure it is error free put a disc in the drive (not the TexBase one) and type the following commands:

```
*SPOOL AMEND1
LIST
*SPOOL
```

This will save the program ready for merging with the original. For safety copy the *Search* program from the TexBase disc onto another one so you will not lose the original program if something goes wrong. Now load the *Search* program from the TexBase disc and put in the disc that you saved *AMEND1* on. Now type:

```
*EXEC AMEND1
```

This will merge the amendments with the original *Search* program. There will be some error messages as a result of the above command but these should be ignored. Now save the completed program on the TexBase disc under the same filename as the original (i.e. *Search*). The complete amended Search program is included on this month's disc as well as the machine code source program.

USING THE PROGRAM

The program is used in the same way as the original except that when starting a search, you are asked whether you want to search in keyword, title or text. Enter 'Y' for any areas in which you wish to search. Searching in title is about as fast as searching in keyword only but searching in the text will slow the program down

considerably due to the large amount of extra information which must be processed. If the words being searched for have been entered as keywords then a search in keyword only should be used since this is far more efficient.

TECHNICAL INFORMATION

The additional coding makes the program very tight on memory space and so little more can be added. I had originally included the assembler source code in the main program but this caused a 'No Room' error. I would like to improve the routine to enter the areas to search but there is too little memory left. Of course, computers with shadow RAM will have plenty of spare memory and cutting the length of variable and procedure names will help.

When searching on text the program loads in each page, converts the text in it to upper case, and searches it for each keyword. If the page matches the search condition then the page is loaded again (so the page is not in upper case) and displays it as normal.

The upper case conversion routine is written in assembly language since it is called many times during a search. It is *much* faster than using string handling routines. It is a good example of what can be achieved with a small amount of simple assembly language.

ted
to a
d to
d. It
e in
nay

```
  105 *L. CODE 900
  106 DIM temp$(7)
 1671 INPUTTAB(20,5);"Search on keyword
";Y$:IFY$="Y" OR Y$="y" THEN keysearch=T
RUE ELSE keysearch=FALSE
 1672 INPUTTAB(20,6);"Search on title ";
Y$:IF Y$="Y" OR Y$="y" THEN titlesearch=
TRUE ELSE titlesearch=FALSE
 1673 INPUTTAB(20,7);"Search on text ";Y
$:IFY$="Y" OR Y$="y" THEN textsearch=TRU
E ELSE textsearch=FALSE
 1674 IF NOT(keysearch OR titlesearch OR
 textsearch) THEN PRINT"NO SEARCH!":A%=I
NKEY(200):CLOSE#0:ENDPROC
 1720 INPUT#Z%,ti$
 1721 P%=P%+154:PTR#Z%=P%
 1722 IF textsearch PROCload
 1723 P%=P%+(lines%*77):O%=-3
 1724 FOR N%=1 TO K%:temp$(N%)=k$(N%):NE
XT:FOR N%=K%+1 TO 6:temp$(N%)="":NEXT
 1725 IF textsearch FOR N%=0 TO lines%:A
$(N%)=FNupper(A$(N%)):NEXT
 1726 PROCcomp
 1780 REM delete this line
 1810 IF keysearch PROCkeysearch
 1811 IF titlesearch PROCtitlesearch
 1812 IF textsearch PROCtextsearch
 1850 REM delete this line
 1880 P%=P%-(lines%*77):PTR#Z%=P%
 1890 PROCload
 1900 P%=P%+(lines%*77)
 1910 CLS:PROCdis
 1920 REM delete this line
 1930 REM delete this line
 1940 REM delete this line
 1950 REM delete this line
 1960 REM delete this line
10000 DEF PROCkeysearch
10010 IFINSTR(kw$,temp$(N%)) >0 yes%=yes
%+1:temp$(N%)="&%&"
10020 ENDPROC
10030 DEF PROCtitlesearch
10040 t$=FNupper(ti$)
10050 IF INSTR(t$,temp$(N%)) >0 yes%=yes
%+1:temp$(N%)="&%&"
10060 ENDPROC
```

```
10070 DEF PROCtextsearch
10080 FOR line%=0 TO lines%
10090 IF INSTR(A$(line%),temp$(N%)) >0 y
es%=yes%+1:temp$(N%)="&%&":line%=lines%
10100 NEXT
10110 ENDPROC
10200 DEF FNupper(m$)
10210 $&920=m$
10220 CALL &900
10230 =$&920
10300 DEF PROCload
10310 FOR N%=0 TO lines%-1
10320 INPUT#Z%,A$(N%)
10330 NEXT
10340 FOR N%=lines% TO 49
10350 A$(N%)=STRING$(75," ")
10360 NEXT
10370 ENDPROC
```

```
  10 REM Program Source
  20 REM Version B 1.0
  30 REM Author  Howard Javes
  40 REM BEEBUG  July 1992
  50 REM Program subject to copyright
```

```
  60 :
 100 FOR pass%=0 TO 2 STEP 2
 110 P%=&900
 120 [OPT pass%
 130 LDX #0
 140 .loop
 150 LDA &920,X
 160 CMP #13
 170 BEQ out
 180 CMP #97
 190 BCC over
 200 CMP #123
 210 BCS over
 220 SEC
 230 SBC #32
 240 .over
 250 INX
 260 JMP loop
 270 .out
 280 RTS
 290 ]
 300 NEXT
 310 OSCLI "SAVE CODE 900 "+STR$~(P%+2)
```

BEEBUG WORKSHOP

Linked Lists

Mike Williams explores the world of linked lists.

Most computer programmers are familiar with the use of arrays. This is a form of data structure through which an element within a group or list of elements can be located by means of its position. Basic, like most high level languages, contains explicit syntax for handling arrays.

For example, if we define an array called *Marks* by writing:

```
DIM Marks(100)
```

then any element, the *i*th element for example, can be referenced with the syntax:

```
Marks(i)
```

Most languages, including Basic, use some form of brackets to denote arrays. However, if we assume for the moment that such a syntax did not exist, then we could implement equivalent ideas using functions and procedures. We could define an array by writing:

```
PROCdefine_array("Marks",100)
```

and we could set an element to a particular value, or read a particular value using:

```
PROCset_element("Marks",i,value)
```

and:

```
value=FNread_element("Marks",i)
```

Since we are only dealing in semantics, we don't need to consider how these procedures and function would be coded. The point that I am trying to make is that the concept of an array is separate to the syntax by which the concept is implemented, and any structure which we choose to devise can be implemented by some means or other. In this and future Workshops we shall be looking at a number of alternative structures where we shall have to write our own implementation. In most cases we shall make use of the simpler array structure already implemented for us in Basic to do this.

LINKED LISTS

In an array, it is the position of an element in an array which determines any order. If a series of random values are stored in an array then the only method of access is the order in which they are stored. If we wanted to access the values in numerical order (or any other order) then the elements would have to be sorted such that the content-related order of the elements corresponded to the array order of elements. No doubt most readers are aware of at least some of the many sorting methods which have been devised to achieve this, such as the popular *bubble sort*, and the highly efficient *quick sort*.

However, if an alternative structure could be devised in which each element was immediately linked to the next (in some perceived order), and that these links were maintained as elements were added to or removed from the set of elements, then the need for sorting would disappear. This is what many

large databases do for efficiency. However, the basic concept is easily rendered more complex if there is a need to access elements in more than one order. This is beyond the scope of what I intend to cover (for the moment at least), but briefly one solution is to create one or more *indexes* each of which allows immediate access to any and all of the elements in a pre-determined order related to some *key* item within each element. Multiple indexes require multiple keys.

Even if this approach obviates the need for sorting large quantities of data, there is still the problem of locating any one element among many. Just as it is time consuming to browse through an array looking for a particular element, so it is equally time consuming following a series of links from one element to another until that being sought is found. Again, indexes can be so organised that a single element can be found quickly, by implementing the index as some form of so-called *tree* structure. More of that in a future workshop.

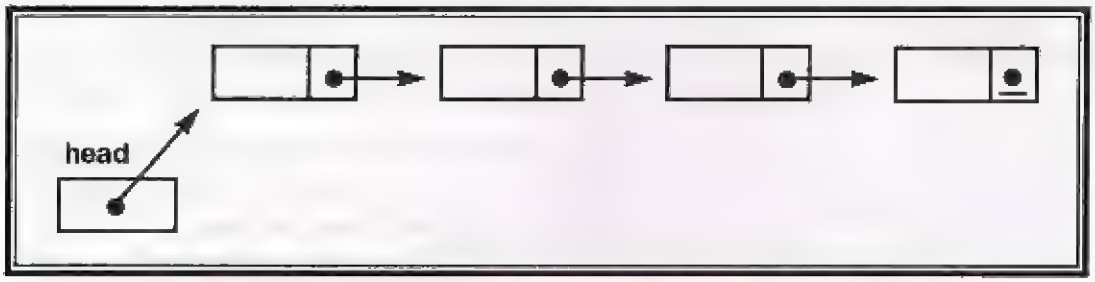


Figure 1. Simple linked list structure showing pointer to head of list and null pointer to terminate list

The simplest form of linked list is shown in figure 1. Each element in the list consists of two items, a data part, and a pointer to the next element. We also need to introduce a *null* pointer, i.e. a pointer which points to nothing and effectively indicates the end of a list, and we need a variable separate to the linked list which points to what is currently the first element in the list. If the list is completely empty then this variable will contain a null pointer.

To implement the list we can use two arrays, one to hold the data, and one to hold the pointers. Let's call them *Data()* and *Link()*. The variable head points to the head of the list, and a second variable free points to another chain of empty elements (initially the entire contents of the array when the list is empty). Null pointers will be indicated by a value of -1. We can now implement a number of functions to enable us to use our linked list:

```
PROCadd_element(data)
FNdelete_element(data)
FNfind_element(data)
FNdisplay_data
```

In order to demonstrate the use of these routines I have added a number of additional procedures to the program which is listed at the end of this month's Workshop:

```
PROCinput_data
PROCdelete_data
PROCfind_data
```

These simply repeat the appropriate function, using a value of 9999 as a terminator. The whole program is controlled by a simple menu system.

For example, when you run the program, select option 1 (*Input Data*) first. Enter as many numbers as you like, pressing Return after each. At any time, enter '9999' to terminate input. Select option 4 and you will see the numbers displayed in order (the order in which they are linked together). Similarly you can also

search for any particular value, and delete selected values. There are a number of safeguards built into the program, to check for searching an empty list, or seeking a non-existent value, for example. However, the program is primarily for demonstration, and not all possible error states are checked for.

In presenting this program now we do not have the space here to discuss its operation in any detail, nor consider some of the other problems that arise. That will have to wait till next month's Workshop.

```
   10  REM Program List01
   20  REM Version B 1.0
   30  REM Author  Mike Williams
   40  REM BEEBUG  July 1992
   50  REM program subject to copyright
   60  :
  100 ON ERROR REPORT:PRINT" at ";ERL:EN
D
  110 DIM Data(500),Link(500)
  120 FOR I%=0 TO 499:Link(I%)=I%+1:NEXT
  130 head=-1:free=0:end%=FALSE
  140 :
  150 REPEAT
  160 MODE7 :choice=FNmenu
  170 IF choice=1 THEN PROCinput_data
  180 IF choice=2 THEN PROCdelete_data
  190 IF choice=3 THEN PROCfind_data
  200 IF choice=4 THEN PROCdisplay_data
  210 IF choice=5 THEN end%=TRUE
  220 UNTIL end%
  230 END
  240 :
 1000 DEF PROCadd_element(data)
 1010 LOCAL L,L1,L2,f%:f%=FALSE
 1020 L=free:free=Link(free):Data(L)=dat
a
 1030 IF head=-1 THEN Link(L)=head:head=
L:ENDPROC
 1040 L1=head:L2=L1
 1050 IF data<Data(L1) THEN Link(L)=head
:head=L:ENDPROC
 1060 REPEAT
 1070   L1=L2:L2=Link(L2)
 1080   IF L2=-1 THEN f%=TRUE ELSE IF da
ta<Data(L2) THEN f%=TRUE
 1090 UNTIL f%
 1100 Link(L)=L2
 1110 Link(L1)=L
 1120 ENDPROC
 1130 :
 1200 DEF FNdelete_element(data)
 1210 LOCAL f%,f1%,L,L1:L=head:L1=L:f%=F
ALSE:f1%=FALSE
 1220 IF L=-1 THEN =L ELSE IF Data(L)=da
ta THEN head=Link(L):Link(L)=free:free=L
:=L
 1230 REPEAT
 1240   L1=L:L=Link(L)
 1250   IF L=-1 THEN f%=TRUE ELSE IF Dat
a(L)=data THEN f1%=TRUE
 1260 UNTIL f% OR f1%
 1270 IF f1% THEN Link(L1)=Link(L):Link(
L)=free:free=L
 1280 =L
 1290 :
 1400 DEF FNfind_element(data)
 1410 LOCAL f%,L:L=head:f%=FALSE
 1420 REPEAT
 1430   IF L=-1 THEN f%=TRUE ELSE IF Dat
a(L)=data THEN f%=TRUE ELSE L=Link(L)
 1440 UNTIL f%=TRUE
 1450 =L
 1460 :
 1600 DEF PROCdisplay_data
 1610 LOCAL L:L=head
 1620 PRINT''"Display Data"
 1630 IF L=-1 THEN PRINT''"List empty":EN
DPROC
 1640 REPEAT
 1650   PRINT Data(L)
 1660   L=Link(L)
 1670 UNTIL L=-1
 1680 PRINT''"Press any key to continue":
L=GET
 1690 ENDPROC
 1700 :
 1800 DEF PROCinput_data
 1810 LOCAL f%:f%=FALSE
 1820 PRINT''"Input Data"
```

```
 1830 REPEAT
 1840   INPUT"Data: " data
 1850   IF data=9999 THEN f%=TRUE ELSE P
ROCadd_element(data)
 1860 UNTIL f%
 1870 ENDPROC
 1880 :
 2000 DEF PROCdelete_data
 2010 LOCAL f%,f1%:f%=FALSE
 2020 PRINT''"Delete Data"
 2030 REPEAT
 2040   INPUT"Data: " data
 2050   IF data=9999 THEN f%=TRUE ELSE f
1%=FNdelete_element(data):IF f1%=-1 THEN
 PRINT data;" not found"
 2060 UNTIL f%
 2070 ENDPROC
 2080 :
 2200 DEF PROCfind_data
 2210 LOCAL f%,f1%:f%=FALSE
 2220 PRINT''"Find Data"
```

```
 2230 REPEAT
 2240   INPUT"Data: " data
 2250   IF data=9999 THEN f%=TRUE ELSE f
1%=FNfind_element(data):IF f1%=-1 THEN P
RINT data;" not found" ELSE PRINT data;"
 found at position ";f1%
 2260 UNTIL f%
 2270 ENDPROC
 2280 :
 2400 DEF FNmenu
 2410 LOCAL c%
 2420 PRINT"LINKED LIST DEMONSTRATION"'
 2430 PRINTTAB(5)"1. Input Data"
 2440 PRINTTAB(5)"2. Delete Data"
 2450 PRINTTAB(5)"3. Find Data"
 2460 PRINTTAB(5)"4. Display Data"
 2470 PRINTTAB(5)"5. Exit"
 2480 PRINT'"Enter 1 - 5:";
 2490 REPEAT:c%=GET-48:UNTIL c%>0 AND c%
<6
 2500 =c%
```

BEEBUG Education (continued from page 25)

the Second World War. One of the best ways to look at conditions experienced by the population at that time is to examine the impact on their day to day lives. One of these was the blackout.

Pupil Sheet 2 (in common with the other four that come as part of the package) suggests the equipment to use. It also has a diagram and a list of resources. It gives a model sequence of steps to take to set up an experiment using a light sensor, operate the software (simple and straightforward), produce a graph and eventually print it out from within the package.

The Teacher's Sheet 2 (as with each Teacher's Sheet) draws attention to those conditions of the experiment which will affect whether it is a fair test or not, whether the results can be judged reliable, and what are the likely pitfalls. There is a template file on disc which will match the instructions and menu options on the Sheets.

There is a video available (not seen for this review) which is aimed at in-service providers, and this shows what Sixth Sense can do, how to do it and how it has been used in the case of two of the sample experiments.

CONCLUSIONS

Both these packages are simple, and are easy to learn and use. They have a specific and well thought out (though - in the case of Style - perhaps somewhat ambitious) slot in the market and are well presented. As such they represent good value for money and do the job for which they were designed very well. It is encouraging that products of this quality are still being developed and that these diverse educational activities are being so well supported at this late stage in the game, as far as the BBC micro is concerned.

ROM Controller

This most useful utility from Andrew Ho, originally published in Volume 5 Issue 3, will help you in looking after your ever-increasing ROM library.

Sideways ROMs have proved to be a particularly popular feature of the BBC micro and Master, and a vast number of users have long ago filled the few available sockets in their machines, and resorted to sideways ROM boards to allow them their full complement of sixteen ROMs.

When you have that many ROMs in your machine however, you really need some kind of ROM management system: something that will allow you temporarily to disable those ROMs causing internal conflicts of command names or workspace.

The ROM controller presented here does just that, and a bit more besides. In precise terms it offers the user the three following features:

1. It will tabulate all ROMs (and sideways RAM) present, giving socket number, name and whether the socket is enabled or not, and it will provide information on ROM type, ROM size and the copyright message.
2. It permits selection and deselection of individual ROMs, with the state of each remaining intact even across a hard Break.
3. It will allow you to save any ROM to disc so that it may be reloaded into sideways RAM. Note that this option does not permit protected software to be run from sideways RAM.

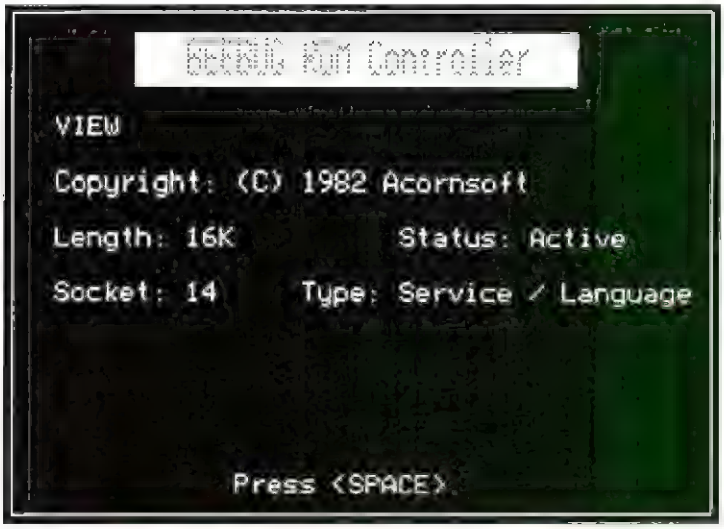
To use the ROM Controller, type in the listing and save it to disc. When it is run, the program will announce itself, and will display the details of each ROM in your machine. It then presents the ROM names in tabular form, with a key legend below. Empty sockets will be labelled accordingly, and any sideways RAM containing no ROM image will be labelled "RAM". Disabled sockets, empty sockets and empty sideways RAM are highlighted in red. You will also notice a cursor against ROM 15, the highest priority socket. This cursor is moved with the up/down cursor keys, and is used to select individual ROMs for the various functions outlined below.

The ROM list

To disable the ROM currently selected (by the cursor position), press 'D', and to indicate this the ROM name will turn red. If, however, you are holding Ctrl at the same time, all the ROMs in your machine will be disabled; and if you hold down Shift, then you will be asked for a wildcard name and then any ROM with that name contained in its title will be disabled. To enable a ROM you have to press 'A' (for Activate), and this works in a similar way to the disabling

procedure. However, before any of the above commands can take effect, you must exit from the controller with the Q option (see below).

To save the currently selected ROM to disc, press 'S'. You will then be asked for its filename (which is automatically truncated to seven characters so as to avoid a 'Bad filename' error on DFS discs) and the required drive number. A null filename aborts the save (i.e. just press Return). The result of a *INFO on the saved file will then be displayed before returning to the main screen.



Information on a particular ROM

To reload a ROM image saved in this way back into sideways RAM is an easy matter. If you have a Master, type:

```
*SRLOAD <filename> 8000 <bank> Q
```

where <bank> is the letter (or number) of the sideways bank to load into, i.e. W, X, Y or Z. If you are using an ATPL board on a BBC micro, just type:

```
*LOAD <filename>
```

When the file has loaded, press Ctrl-Break to allow the ROM to claim its workspace. If you are using a BBC micro and your sideways RAM is installed in another make of ROM board, you may need to refer to the manufacturer's instructions to find out how to load ROM images to RAM.

To obtain further information about a selected ROM press 'I'. This gives the full copyright message together with the ROM type - Service, Language or both - the length of the ROM and whether it is currently enabled or not.

Pressing 'O' from the main screen allows star commands to be performed while 'Q' will quit the routine in a proper manner, and set the on/off status of each ROM, as currently selected.

MEMORY USAGE

As the program stands, the machine code sections are assembled into an area of memory at &A00, and the oscli routine uses a few bytes at &900 for passing commands to the operating system. Neither of these areas should cause any conflict. If you do wish to alter the locations used however, you will need to modify lines 1010, 1390 and 1890.

PROCEDURES

assemble	Sets up the machine code routines for downloading parts of the ROMs, and also OSCLI commands.
swrst	Tests if sideways RAM is present and if so tests to see if there is ROM software in it.
list	Prints out the names of the ROMs.
instr	Prints out the key controls.
cursor	Prints string beside the ROM currently chosen.
os	Performs the OS commands specified in Q$.
break	Sets up the BREAK intercept routine for disabling ROMs through Ctrl/Shift/soft/memory clear BREAK.
status	Prints out all information about the ROM currently selected (length, type, name, copyright,

	active/disabled, & socket number).
oscli	This allows the user to perform any OS command.
disable	This tests if Shift is being pressed and if so goes to the wildcard disable procedure. If Ctrl is being pressed it goes to the universal disable procedure, otherwise the currently selected ROM is disabled.
enable	This does exactly the same as the disable procedure, but enables ROMs instead of disabling them.
alloff	Disables all ROMs.
allon	Enables all ROMs, while not enabling empty sockets.
wcoff	Asks for a wildcard name, and then disables all ROMs containing that wildcard name.
inputname	Input procedure for the two wildcard procedures.

VARIABLES

A%	Number of currently selected ROM.
AC%()	Current codes in the ROM table for each ROM.
A$	Character used as pointer (" " for deleting old pointer and ">" for pointer).
C	Y co-ordinate of pointer (">").
C$()	Copyright messages of ROMs.
DR$	Drive number.
F$	Filename of ROM to be saved.
L%()	Lengths of ROMs (8/16 K).
N$()	Names of ROMs.
O$	OS command entered by user in PROCoscli.
Q$	OS command to be carried out.
R$	Wildcard name.
SP%	Sideways RAM: 0 if not present, 1 if present.
SU%	Sideways RAM in use: 0 if not in use, 1 if in use.
T%()	ROM types.

```
   10 REM Program ROM Controller
   20 REM Version B 0.4
   30 REM Author  Andrew Ho
   40 REM BEEBUG  July 1992
   50 REM Program Subject to Copyright
   60 :
  100 ON ERROR GOTO410
  110 MODE 7:VDU23,1,0;0;0;0;
  120 FORL%=0TO1:PRINTTAB(5,L%);CHR$131;
CHR$157;CHR$129;CHR$141;"BEEBUG ROM Cont
roller"SPC(3);CHR$156:NEXT:VDU28,0,24,39
,3
  130 PROCassemble:PRINT;CHR$134;"Initia
lizing... ":?&275=1:PROCos("FX4,2")
  140 DIM N$(15),C$(15),AC%(15),L%(15),T
%(15)
  150 FOR A%=0 TO 15:VDU31,17,0:PRINT;A%
:?&66=A%:?&60=&80:?&61=&3C:?&62=1:CALL c
opy:?&60=&A0:?&61=&5C:?&62=1:CALL copy
  160 IF !&3C09=&80808080 OR !((?&3C07)+
&3C00)<>&29432800 N$(A%)="--- Empty ---"
:T%(A%)=0:L%(A%)=0:C$(A%)="":AC%(A%)=0:P
ROCswrtest:GOTO250
  170 B%=&3C09:REPEAT:IF ?B%>31 N$(A%)=N
$(A%)+CHR$?B%
  180 B%=B%+1:UNTIL ?B%=0 OR B%=&3C29
  190 T%(A%)=?&3C06
  200 B%=(&3C00+?&3C07):REPEAT:IF ?B%>31
 C$(A%)=C$(A%)+CHR$?B%
  210 B%=B%+1:UNTIL ?B%<10
  220 D$="":B%=&5C09:REPEAT:IF ?B%>31 D$
=D$+CHR$?B%
  230 B%=B%+1:UNTIL ?B%=0 OR B%=&5C29
  240 IF D$=N$(A%) L%(A%)=8 ELSE L%(A%)=
16
  250 AC%(A%)=?(&2A1+A%):NEXT
  260 VDU23,1,0;0;0;0;:PROClist:C=0:PROC
cursor(">")
  270 REPEAT:L=INKEY(5)
  280 IF INKEY(-58) AND C>0 PROCcursor("
 "):C=C-.5:PROCcursor(">")
  290 IF INKEY(-42) AND C<15 PROCcursor(
" "):C=C+.5:PROCcursor(">")
  300 IF INKEY(-17) GOTO 370
  310 IF INKEY(-82) GOTO 900
  320 IF INKEY(-38) PROCstatus
```

```
  330 IF INKEY(-55) PROCoscli
  340 IF INKEY(-51) PROCdisable
  350 IF INKEY(-66) PROCenable
  360 UNTIL FALSE
  370 CLS:?&275=0:PROCos("FX15"):PROCos(
"FX4")
  380 PROCbreak
  390 END
  400 :
  410 ON ERROR OFF:ON ERROR GOTO 410
  420 CLS:PRINT':REPORT:PRINT" at line "
;ERL
  430 PRINT"Press Spacebar to continue:"
;:REPEATUNTILGET=32
  440 GOTO260
  450 :
  460 DEFPROCswrtest
  470 IF FNtestswr(A%)=1 N$(A%)="<RAM>"
  480 ENDPROC
  490 :
  500 DEFFNtestswr(Q%):Y%=Q%:!&F6=&8000:
!&70=USR&FFB9
  510 ?&71=(?&70)+1:CALLtestit:IF?&71=?&
70 =0 ELSE =1
  520 :
  530 DEF PROClist
  540 CLS:FOR A%=15 TO 0 STEP -1:VDU136,
32,137:IFA%<10 PRINT;" ";
  550 PRINT;A%;:IF AC%(A%)=0 PRINT" ";CH
R$129;N$(A%) ELSE PRINT" ";CHR$131;N$(A%
)
  560 NEXT:PROCinstr:ENDPROC
  570 :
  580 DEF PROCcursor(A$)
  590 PRINTTAB(1,C)A$
  600 ENDPROC
  610 :
  620 DEF PROCoscli
  630 PROCos("FX15"):PROCos("FX138,0,127
")
  640 CLS:INPUT"Command: *"O$
  650 PROCos(O$)
  660 PRINT'SPC(12);"Press <SPACE>.":REP
EAT UNTIL GET=32:PROClist:PROCcursor(">"
)
  670 ENDPROC
  680 PRINTSPC(8);:REPORT:GOTO 660
  690 :
  700 DEF PROCROMoff
  710 FOR A%=15 TO 0 STEP -1:AC%(A%)=0:?
(&2A1+A%)=0:VDU31,6,A% EOR 15,129:NEXT
  720 ENDPROC
  730 :
  740 DEF PROCROMon
  750 FOR A%=15 TO 0 STEP -1:AC%(A%)=T%(
A%):?(&2A1+A%)=T%(A%):IF AC%(A%)<>0 VDU3
1,6,A% EOR 15,131
  760 NEXT:ENDPROC
  770 :
  780 DEF PROCstatus
  790 CLS:A%=C EOR 15:PRINT;CHR$131;N$(A
%)''CHR$131;"Copyright:";CHR$133;LEFT$(C
$(A%),27);
  800 IF LENC$(A%)>27 PRINT;SPC(12);CHR$
133;MID$(C$(A%),28,27);CHR$11
  810 PRINT''CHR$131;"Length:";CHR$133;L
%(A%);"K";SPC(9);CHR$131;"Status:";CHR$1
33;
  820 IF AC%(A%)<>0 PRINT;"Active" ELSE
PRINT;"Disabled"
  830 PRINT'CHR$131;"Socket:";CHR$133;A%
;SPC(4);CHR$131;"Type:";CHR$133;
  840 IF (T%(A%) AND 128)=128 PRINT;"Ser
vice";
  850 IF (T%(A%) AND 128)=128 AND (T%(A%
) AND 64)=64 PRINT;" / ";
  860 IF (T%(A%) AND 64)=64 PRINT;"Langu
age"
  870 PRINT''''SPC(12);"Press <SPACE>."
:PROCos("FX15")
  880 REPEAT UNTIL GET=32:CLS:PROClist:P
ROCcursor(">"):ENDPROC
  890 :
  900 CLS:VDU23,1,1;0;0;0;:A%=C EOR 15:P
RINT;CHR$131;N$(A%):?&66=A%:?&60=&80:?&6
1=&3C:?&62=(L%(A%)*4):CALL copy:PROCos("
FX15")
  910 PRINT'CHR$131;"Filename:";CHR$133;
:INPUT""F$
  920 PRINT'CHR$131;"Drive:";CHR$133;:IN
PUT""DR$
  930 PROCos("DRIVE "+DR$)
```

```
 940 F$=LEFT$(F$,7)
 950 PROCos("SAVE "+F$+" 3C00+"+STR$~(L
%(A%)*&400)+" D9DC 8000")
 960 PRINT';CHR$11:PROCos("INFO "+F$)
 970 PRINT'''SPC(4);CHR$133;"Press any
 key.";CHR$11;CHR$13:G=GET:VDU23,1,0;0;0
;0;
 980 CLS:PROClist:PROCcursor(">"):GOTO
270
 990 :
1000 DEF PROCos(Q$)
1010 $&900=Q$:CALL oscli:ENDPROC
1020 :
1030 DEF PROCdisable
1040 IF INKEY(-2) PROCROMoff:ENDPROC
1050 IF INKEY(-1) PROCalloff:ENDPROC
1060 A%=C EOR 15:AC%(A%)=0:VDU31,6,C,12
9:?(&2A1+A%)=0:ENDPROC
1070 :
1080 DEF PROCenable
1090 IF INKEY(-2) PROCROMon:ENDPROC
1100 IF INKEY(-1) PROCallon:ENDPROC
1110 A%=C EOR 15:IF AC%(A%)=0 AND T%(A%
)<>0 AC%(A%)=T%(A%):VDU31,6,C,131:?(&2A1
+A%)=T%(A%)
1120 ENDPROC
1130 :
1140 DEF PROCbreak
1150 FOR I%=0 TO 2 STEP 2:P%=&132:[ OPT
I%
1160 LDY#0:.loop:LDAtable,Y:STA&2A1,Y:I
NY:CPY#16:BNEloop:RTS
1170 .table:]:FORU%=0TO15:P%?U%=U%?&2A1
:NEXT:P%=P%+16
1180 NEXT:!&287=&01324C:ENDPROC
1190 :
1200 DEF PROCinstr
1210 PRINT;CHR$133;"<Q>uit";SPC(14);"<I
>nfo"'CHR$133;"<O>scli";SPC(13);"<S>ave"
'CHR$133;"<A>ctivate";SPC(3);CHR$135;"(+
CTRL  = All ROMs"'CHR$133;"<D>isable";SP
C(4);CHR$135;"(+SHIFT = Wildcard":ENDPRO
C
1220 :
1230 DEF PROCalloff
1240 PROCinputname
1250 FOR C=0 TO 15:A%=C EOR 15:IF INSTR
(N$(A%),R$)<>0 PROCdisable
1260 NEXT:?&259=0:C=?&70:ENDPROC
1270 :
1280 DEF PROCallon
1290 PROCinputname
1300 FOR C=0 TO 15:A%=C EOR 15:IF INSTR
(N$(A%),R$)<>0 PROCenable
1310 NEXT:?&259=0:C=?&70:ENDPROC
1320 :
1330 DEF PROCinputname
1340 ?&70=C:VDU28,0,24,39,20,12
1350 PROCos("FX15"):INPUT"Wildcard name
: "R$
1360 ?&259=1:CLS:PROCinstr:VDU28,0,24,3
9,3:ENDPROC
1370 :
1380 DEF PROCassemble
1390 FOR I%=0 TO 2 STEP 2:P%=&A00:[ OPT
I%
1400 .test
1410 LDA &F4
1420 STA &65
1430 LDA #&F
1440 STA &F4
1450 STA &FE30
1460 LDY #0
1470 .tloop
1480 LDA &8000,Y
1490 STA &70,Y
1500 INY
1510 CPY #16
1520 BNE tloop
1530 LDA &65
1540 STA &F4
1550 STA &FE30
1560 RTS
1570 .copy
1580 LDA &60
1590 STA &A9
1600 LDA #0
1610 STA &A8
1620 STA &AA
1630 LDA &61
1640 STA &AB
```

Continued on page 48

To celebrate the tenth anniversary of the founding of BEEBUG magazine, we have put together a selection of the best programs which we have published over the past ten years. This disc is packed with applications, utilities and games most of which have not been available previously other than when first published in BEEBUG. All the programs come with full on-screen help files, which can be printed out as well for permanent reference.

BEEBUG 10th Anniversary Disc

Celebrate BEEBUG's 10th anniversary with this disc at the special low price of £4.95 and you will have something to celebrate too.

DRAUGHTS - An implementation of the classic board game in which you pit your wits against a computerised opponent.

KEYSTRIP DESIGNER - A very well written program to enable the creation, editing and printing of function key strips.

GARP - GARP (Geographical Atlas using Radial Projection) allows views of the globe to be displayed from any point above the Earth's surface.

MULTI-COLUMN PRINTING - This utility formats any text file into columns, and prints the result using an Epson FX-80 or compatible printer.

PERPETUAL CALENDAR - This program can display or print the calendar month by month for any year between 1753 and 5000 A.D. in the United Kingdom, or even earlier in other countries.

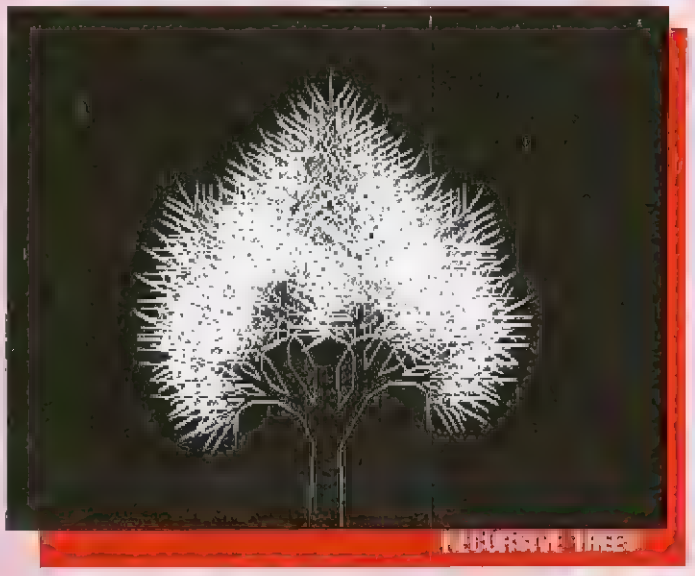
QUAD - Quad is a Tetris-like game, in which you must manipulate falling blocks to slot into each other. Dangerously addictive:

STEALTH - In this game you play against an opponent (or the computer), who sets a number of targets for you to find, and you must use your skill to discover the ocations of the targets in as few goes as possible.

RECURSIVE TREES - This fascinating program uses recursion to create an infinite variety of tree-like designs - you choose a set of numbers, and the computer does the rest.

THE WORLD BY DAY AND NIGHT - This program will draw a map of the world showing graphically where the sun is in the sky or where it's night at every spot on earth.

CROSSWORD COMPILER - This program allows crosswords to be designed and the clues compiled.



PBB3a	*3.5" ADFS*	*£4.95 inc. VAT + £1 p&p*
PBB5a	*5.25" DFS 40/80T*	*£4.95 inc. VAT + £1 p&p*

RISC Developments Limited 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. (0727) 40303 Fax. (0727) 860260

Mr Toad's Machine Code Corner 4

A pond full of ramblings from Mr Toad.

Hi there, Toad fans! A request for your opinions this month - how do you personally, pronounce 6502 assembler in your own mind? How do you read it out to someone else? Most of the mnemonics will just be spoken as three letters, of course: Jay Es aR, Bee eN Ee, etc. Not all, though. I expect you say 'jump' for JMP, naturally. What about CMP? Do you say 'compare' or Cee eM Pee?

It's the other bits that interest Mr T: do you say 'ten hex', 'hex ten', 'one oh hex', or what? Clearly, one ought not to say 'ten' for &10, but I bet you do! The hardest one is the hash. How do you distinguish vocally, or sub-vocally, between #&FE and &FE? This one is a prime cause of bugs with Mr T. In fact, a lot of bugs probably stem from inconsistencies in such matters when reading listings to oneself.

Mr T - like most of you, probably - has nobody to talk to about the arcane topic of assembly-language. Mrs T and the tadpoles go wibbly if it is so much as mentioned. Having learnt it all from books and magazines, Mr T has never heard anyone else say the words. There's bound to be an 'official' version in a book somewhere, but if it's a book on the BBC, Mr T has missed it. Write in with your views.

On this subject, did you realise that the assembler mnemonics are not tokenised by the Basic assembler? Those which come from Basic are - AND, DIV, MOD and so on. ORA is special, being stored as &84 (the token for OR) followed by &41 (ASCII "A"). The real assembler mnemonics are just held as three letters. Mr T thinks it is a crying waste of space - two bytes per instruction add up to a lot when you're desperate. Could it really not have been done differently? After all, Basic "knows" it's into assembly text after the opening bracket, so all sorts of Basic words like PRINT, DIM, IF, etc. could have had their tokens duplicated. I should write to my MP, if I were you.

Going back to the pronunciation of 6502 assembler, why not make it double up as literature? With ingenuity it should be possible to write great poetry which also runs as meaningful code. It's been done with chemistry: here's a very old one. I'll leave you to work out how to declaim it.

There was a young Chemist called Hall
Who made a hexagonal ball,
Its molecular weight
Was $\frac{\text{COS } \theta * 8}{x + \sqrt{0}}$

Toads being highly cultured amphibians - as I never cease to remind you - Mr T has decided to hold a competition (a real one, for once), for poetry couched entirely in 6502. It must scan and rhyme - none of yer free verse for Auntie Acorn. A few liberties will be allowed; the final RTS may be omitted and you can make it just part of a working routine. Line numbers, colons and brackets may be silent.

Here's a Limerick by way of illustration:

```
10 osasci=&FFE3
20 FOR n=0 TO 2 STEP 2
30 P%=&DD00
40 REM Limerick starts
50 :
60 CLS
70 [
80 OPT n
90 .printOutText
100 LDY #&2F
110 .andTheNext
120 LDA toadText,Y
130 JSR osasci
140 DEY
```

```
150 BNE andTheNext
160 :
170 \ Limerick ends
180 RTS
190 .toadText
200 EQUB &0D
210 EQUS ")sdrawkcab( dnuor og dlrow
eht sekam rewoP daoT"
220 EQUB &0C
230 ]
240 NEXT
250 CALL printOutText
```

This being truly avant-garde verse, the string .toadText is printed out backwards. Send in your masterpieces, envelopes marked "MCC comp" on the outside, and I will include the best offerings (if not obscene) in a future column. The winner will get a large badge, hand-decorated by the tadpoles, bearing the legend "I'M A SWOT".

Following the bit about tunes the other month, I have received a letter from none other than Fergal McPhurgle, Laird of Glen Phurgle and hereditary Chief of the Clan McPhurgle. In case any ignorant sassenachs should be reading this, I will point out that he should be referred to as 'The McPhurgle' and addressed simply as 'Phurgle', except by very important people, who may call him Fergal. My reply to his query, which I hope concerned bagpipes, is as follows.

Weel, Fergal, ye cuid fix an auld vacuum-cleaner tae the moothpiece wi' a wee microswitch tae cut it off as the bag inflates. Ye cuid control it wi' the cassette relay. Ye'd have tae puit wee solenoids over the holes in the chaunter, controlled via the user port. The software shouldnae be tae deeficult. What wuid ye dae aboot the drones, though?

That's about it for this month, Toad fans. Next month we help Oxfam by designing 6502-controlled hamster-wheel generators for the third world.

ROM Controller (continued from page 45)

```
1650 LDA &F4
1660 STA &65
1670 LDA &66
1680 STA &F4
1690 STA &FE30
1700 LDX #0
1710 .cloop1
1720 LDY #0
1730 .cloop2
1740 LDA (&A8),Y
1750 STA (&AA),Y
1760 INY
1770 BNE cloop2
1780 INX
1790 INC &A9
1800 INC &AB
1810 CPX &62
1820 BNE cloop1
1830 LDA &65
1840 STA &F4
1850 STA &FE30
```

```
1860 RTS
1870 .oscli
1880 LDX #0
1890 LDY #&09
1900 LDA #0
1910 JSR &FFF7
1920 RTS
1930 .testit
1940 LDX&F4
1950 STY&FE30
1960 STY&F4
1970 LDA&71
1980 STA &8000
1990 LDA &8000
2000 STA&71
2010 LDA&70
2020 STA &8000
2030 STX&F4
2040 STX&FE30
2050 RTS
2060 ] NEXT:ENDPROC
```

Direct Memory Access (3)

by Alan Wrigley

1st course

In the first two articles in this series on direct memory access, we have looked at the three indirection operators which allow us to write to and read from memory locations directly, and we have considered some of the areas of memory that you might want to treat in this way. We also saw that in some cases you may want to access specific memory locations explicitly, while at other times you will ask Basic to reserve an area for you, which you then access by reference to a variable without necessarily knowing the actual addresses you are using.

We will now look in detail at some specific uses for direct memory access.

MACHINE CODE UTILITIES

Sometimes a Basic program will load a machine code utility to perform a particular task, or it may assemble and run such a utility from within the program itself. Machine code programs can also perform tasks independently of any other program, and in this case are loaded and run directly from the command prompt by the user.

There are many things which are better done by a machine code program than by a Basic one, and indeed many that can *only* be done in machine code. Many examples of this kind of program have been published in BEEBUG over the years - some recent examples are the Mode 0 Screen Dump in Vol.11 No.1, the Disc Spooler Utility in Vol.10 No.9, and the Basic Error Exposer in Vol.10 No.8. All three of these could be run directly from the command prompt or from within a Basic program, depending on the circumstances in which the utility is required.

If the utility is loaded from disc by the Basic program, then clearly it must exist separately from the calling program, and thus will have been created previously. This is usually done by running a Basic source program which assembles the machine code and saves it to a file. All three of the utilities mentioned above are created in this way. It is not the purpose of this article to explain machine code programming - there are books available on the subject if you are interested to find out more.

When the utility exists separately like this, it is usual to use a specific area of memory into which to load it. The start address of this memory block is then used as the address at which the code is assembled and is thus built into the source program. This address is saved with the utility, and in future when it is loaded using the *LOAD command, or run using *RUN, it will be loaded automatically at that address unless another is specified at the time of loading. For example, if a utility *MyUtil* has been assembled and saved with a start address of &900, it can be loaded at &900 by:

```
*Load MyUtil
```

while the following command will load it at &900 and also run the code at that address (unless a different execution address was specified when the file was saved):

```
*Run MyUtil
```

We considered some areas of memory which could be used for this purpose in last month's article. Among the most popular are pages 9 and 10 of the memory map - i.e. locations &900-&9FF and &A00-&AFF. You will see that the three utilities we have mentioned all choose &900 as a suitable location. If you read the article on Mode 0 Screen Dumps in Vol.11 No.1 you will also see that on a Master you could use memory from &DD00-&DEFF. We have not mentioned this area before - it is part of the Master's private RAM and is

not available on the model B. It is known as *Transient utility workspace* and is actually provided for the purpose we are discussing here (though it is also used by the *MOVE command, so if this command is issued it will corrupt any utility loaded there).

A machine code program which is assembled to run at a particular address in this way should always be loaded at that address. It is very unlikely that it will run from any other address, since there will almost certainly be references to specific locations within the program.

As an alternative to creating a separate machine code utility and then loading it, a Basic program may simply assemble a piece of code within the main program and call it when required. In this case it is not necessary for the code to exist as a file - the source is part of the Basic program and the machine code only exists temporarily when the program is run. It is perfectly possible to use specific areas of memory for this purpose, such as pages 9 or 10 as we have already seen, but generally it is easier to reserve a block of memory using DIM. The variable specified in the DIM statement will then hold the start address of the code, and this can be used with the Basic CALL statement to run the code.

So far in these examples we have not made any use of the indirection operators. This may not always be necessary, but there are many instances where a Basic program needs to interact with a machine code utility assembled from within itself. Information may need to be transferred in either direction, and in these cases you would need to use the indirection operators. For example, suppose that a piece of code carries out a mathematical calculation on a four-byte integer, and returns the result to the Basic program. The simplest way to do this is to include four bytes at the end of the piece of code, which are not part of the actual program code, with a label to identify the address. The Basic program then accesses these using the pling operator, while the machine code section can load from and store to the address directly.

For example, assuming that the label is called *store*, and the value to be processed is held in *val%*, the Basic program would pass the value to the utility, and print the result, in the following way:

```
!store=val%
CALL code%
PRINT !store
```

SCREEN ACCESS

All the information that appears on your screen, whether text or graphics, is held in an area of memory set aside for the purpose. In the case of mode 7, a series of memory locations holds the ASCII value of each character displayed, reading from the top left. When you write to the screen, using for example a statement such as:

```
PRINT "A"
```

then the contents of the location which holds the character at the cursor position will change, in this case to 65 (the ASCII code for A). The screen memory in mode 7 starts at &7C00, so you can verify the above with the following simple program:

```
10 MODE 7
20 PRINT "A"
30 PRINT ?&7C00
```

Below the "A" you will see the value "65" on the screen, printed by line 30. The MODE statement at line 10 ensures that the screen is cleared, so that the "A" will be printed at the very top of the screen (i.e. at location &7C00).

In other modes, the situation is similar but rather more complex, since in this case the screen is divided up not into character positions but pixels. This is what enables text and graphics to be mixed on the same screen. Characters are thus held in memory not as ASCII codes but as a set of pixels making up the graphic representation of the character.

What this means in practice is that you can write directly to the screen by

accessing the memory locations required. Normally you would not need to do this, since BBC Basic has a full set of commands to place both text and graphics on the screen using the VDU drivers which are built into the operating system. These take care of all output to screen or printer, and are largely transparent to the user. But there are cases when direct screen access can be an advantage. Speed is the main reason - direct access is much faster than the VDU drivers - and so you will often find this technique used by games. Another use for direct access is to save a screen directly to disc and to load it back in by the same route - again, games often load a title screen in this way.

You can use this facility to good effect in your own programs; for example, you could design a screen full of graphics, and save it with the command:

```
*Save MyScrn aaaa bbbb
```

where aaaa is the start address of screen memory in hex (normally the same as HIMEM) and *bbbb* is the end of screen memory, also in hex (normally 8000). Note that you must be using a non-shadow screen mode to be able to do this. Provided that your program is operating in the same mode as the screen which was saved, you can then load it back in at any point with:

```
*Load MyScrn
```

DATABASE RECORDS

Handling fixed-length records in a database is a very common use of direct memory access. Let's take a simple example. Suppose that you are writing a database which is designed to hold records consisting of a specific number of fixed-length fields - perhaps an address book or something similar. You could of course handle the information in each record by using variables; for example, you could have a variable *name$* which holds the name, an array *address$* with perhaps 5 elements to hold five lines of address, and so on. But often a much neater way is to dimension a block of memory equal in length to one record, load the required record into that block in one go, and process the data using the indirection operators. This is easy to do when you are working with fixed-length fields, since the start of each field relative to the start of the record will always be the same.

Thus if the name field is 20 bytes long, each address line is 25 bytes, and a block of memory is reserved at *record%*, then you can access the name by using:

```
$record%
```

followed by the first line of the address with:

```
$(record%+21)
```

and the second line with:

```
$(record%+47)
```

and so on. You will see that each string is one byte longer than the corresponding field, as the dollar operator adds or expects a carriage return (at the end of the string) when writing or reading data. You can also include numeric fields just as easily. Suppose that you wanted a field for age located between the name and address fields. This could be accommodated by a four-byte integer, and so could be accessed using:

```
record%!21
```

In this case no carriage return is necessary, so the first line of the address would now start at *record%+25*.

As well as avoiding a clutter of variables and arrays, this approach has a couple of advantages. Firstly, it is possible to load and save a complete record in one go, rather than using PRINT or INPUT for each variable in turn. Secondly, if you are handling only text fields, the data in the file can be read by a text editor or transferred more easily to other computer systems. This is because if you are filing strings from Basic using PRINT, they are filed in a slightly coded format which can only be read by the INPUT statement.

There are many more uses for direct memory access; this series of articles has hopefully given you an insight into some of the possibilities. In next month's *1st Course* we will move on to a new topic.

512 Forum

by Robin Burton

This month we continue with last month's topic by looking further at CHKDSK, the DOS command for checking the integrity of discs.

Last month we got as far as DOS disc organisation, some of the problems that can occur and the fact that CHKDSK might be able to recover data after certain types of disc corruption. Now we'll begin to look at the various options the program offers in these cases.

CHKDSK SWITCHES

As well as the ability to check the integrity of a disc's data (when no switches are set), plus the verbose (/V) switch option which provides a step by step display of what CHKDSK is doing as it runs, there are five more switches. These are of varying complexity in what they do and each of them might prove useful in certain circumstances.

The switch settings (except /V, mentioned last month) are shown below with a brief explanation of their meaning. Some are for more immediately obvious purposes than others, but unfortunately in spite of this the Digital Research DOS Plus User's Guide doesn't offer much illumination for the more obscure functions. It seems the author assumed that either you'd already know how to use the options, or you're one of those people who doesn't need or want to know.

This leaves most users with the classic 'chicken and egg' paradox. How do you to start to use something for the first time if you're supposed to know about it before you use it and the manual doesn't explain? I don't subscribe to that approach, so we'll examine each of the CHKDSK options in detail.

However, I should warn you that if you need to use some of these functions you may also have to be prepared to do a bit of disc editing during the process. Here are the remaining switches, with a brief description of their functions:

/F Fix errors
/B Check for bad blocks
/D Locate directories
/L Link clusters
/R Recover root directory directories

INITIAL CONSIDERATIONS

Before we examine the switches though, I must make another point. When you embark on any sort of fix to a damaged disc, it's presumably implicit that you don't have a reliable backup, otherwise you wouldn't need to be doing it.

That being the case, you should always take a copy of the faulty disc if you can. It might seem odd to suggest backing up a corrupted disc, but doing so is the only way to ensure that you can get back to a known situation if something goes disastrously wrong during the fix process. Without this safety net you might very well end up in a worse position than that you started from.

For example, if you're using DOS floppies you might be able to use the DISK command to copy the whole disc, including the errors, to a new disc before you proceed. If it's a 640K disc you can

alternatively use ADFS facilities instead if you prefer. However, neither of these options will work if the problem is a bad sector or a bad track rather than a logically corrupted FAT or directory.

In the case of physical rather than logical disc corruption, such as a missing sector or a CRC error, your choices are more limited, but there might still be a few options. You could use a BBC program like ACP's Advanced Disc Investigator ROM to make an exact duplicate of the disc (including bad sectors/tracks) if you have such software.

You might even be able to fix some errors, like CRC problems or missing sectors, with such tools, as I have done several times. If this sort of approach isn't open to you all you can do is recover whatever files you can read, cross your fingers and hope for the best (at the same time taking an oath to maintain better backups in future).

For winchester users, copying an entire disc is obviously not an option, but if damage is limited you might still be able to save some of the files before you start. It should go without saying that if you have good up-to-date backups of all your files you can avoid most of this sort of trouble in the first place, but unfortunately good backups don't necessarily guarantee complete protection against all problems.

If you're a frequent 512 user like me, the 'Law of Sod' will inevitably take over at some point; you know the one - "anything that can go wrong eventually will". When it does, the new file you've just been working on will be the one that's lost. This has happened to me on more than one occasion and, being unaware of any problem after a seemingly normal save when finishing work, I've even taken a backup, also apparently completely successfully, and then switched the machine off.

Everything seems fine and most of the time it is, but on a few occasions I've found that the next time the 512 is switched on, my original new file is corrupted. What's worse, since the backup file is a faithful copy, it is a perfect copy of a corrupted file and is therefore useless. Obviously, although the file save seemed normal at the time, it wasn't. This is where CHKDSK might be your only salvation.

FIX ERRORS

The first switch in the list, /F, can be appended to any CHKDSK command in the way shown for the /V switch last month. /F is usually used after you've established the nature of a disc problem and you've decided that CHKDSK is an appropriate way to try to fix it, perhaps after some manual editing too.

In other words, the /F switch tells CHKDSK to try to fix the trouble it finds, but as mentioned last month, if you don't supply the /F switch, even if disc errors are found by CHKDSK it won't update the disc.

This can cause a moment's panic for new users. If it finds errors CHKDSK always tells you (if the /F switch wasn't specified), but it still asks if you'd like the faults corrected anyway, even if the /F switch wasn't set. It's a simple 'Y/N' choice, but the worrying part is that it looks at first sight as if the disc might be updated whether you asked for it or not. In fact you can relax, as without the /F switch no update takes place no matter what, so I regard this

unnecessary question as a bug. Without the /F switch the question shouldn't be asked, but as you can see in our first example below, it is.

To demonstrate CHKDSK I created a 360K disc containing three example files, FILE1, FILE2 and FILE3, in a single subdirectory called SUBDIR1 (great names, huh!). I've used this disc as the basis for the examples since a bigger disc, more files or more directories would produce too lengthy a display.

For each fault I then manually corrupted the FAT and the directory in various ways to show several of the most common errors CHKDSK is likely to report.

LOST CLUSTERS

The first example shows the CHKDSK report after FILE3's directory entry was deleted, simply by editing E5h (i.e. E5 in hex) into the first character of its name in the subdirectory. This, so far as DOS is concerned, means that the file entry is gone, but the occupied clusters in the FAT remain allocated although they don't 'belong' to any file. This is the sort of thing that could result from a failure during copying or writing a new file.

The situation is referred to either as *lost clusters*, since nothing points to or owns the clusters but they're not free, or alternatively it's sometimes called *loose clusters*, because there are allocated clusters in the FAT but they don't belong anywhere. Which name you prefer depends on which way you look at it, but the two mean the same.

This situation can occur when there's a power glitch or failure during a disc update after writing the data and rewriting the FAT, but before updating the directory concerned. In this case the FAT and the directory will obviously not agree. The initial CHKDSK report for our example looks like this:

Errors found, F parameter not specified.
Corrections will not be written to disk.

4 lost clusters found in 1 chains.
Convert lost chains to files (Y/N)? n

4,096 bytes disk space
would be freed

362,496 bytes total disk space
1,024 bytes in 1 directories
8,192 bytes in 2 user files
353,280 bytes available on disk

Note the prompt requiring an answer. It doesn't matter whether it's Y or N (upper or lower case) in this case, but you must reply.

If CHKDSK is now run again with the /F switch set these loose clusters will be allocated to a filename provided by CHKDSK. You can then rename, copy or load that file and hopefully rebuild your original file. The display for this is shown below:

4 lost clusters found in 1 chains.
Convert lost chains to files (Y/N)? y

362,496 bytes total disk space
1,024 bytes in 1 directories
8,192 bytes in 2 user files
4,096 bytes in 1 recovered files
349,184 bytes available on disk

In this case we know that these four clusters (each of two 512 byte sectors) originally belonged to FILE3 which was in SUBDIR1, but CHKDSK can't be expected to know that and so the newly created file is always put into the root

directory, the display for which in this case now appears as:

```
Volume in drive A has no label
Directory of A:\

SUBDIR1      <DIR>  14-05-92    12:31
FILE000B CHK      4096
       2 File(s)     349184 bytes free
```

showing our original lone subdirectory and the new file created from the loose clusters. The last four hex digits in the filename change for each file created, but this isn't significant, while the extension is intended to remind you that this is a CHKDSK created file.

OTHER FACTORS

Of course it's quite possible in practice that loose clusters which originally belonged to several files will be found on a corrupted working disc. Equally, it's by no means certain that all the clusters belonging to each file will conveniently be located in a single contiguous block of clusters on the disc, nor even that all the linked clusters in a chain have remained intact either.

This is something else CHKDSK can't know about, so recovery isn't always as simple as the artificial example here. The best you can expect from CHKDSK under normal circumstances is that each single chain of linked loose clusters will produce one new temporary CHKDSK file. In consequence a very fragmented disc with a corrupt FAT repaired by CHKDSK might very possibly produce more (temporary) files than you originally started with. Of course each new CHKDSK file in that case might well be a mixture of clusters from several of your original files too, if corruption was severe. As a result, once you've made the FAT and the directories consistent using CHKDSK, the rest is up to you.

Again you should begin by backing up the disc before you try any recovery action. For text files the next step is to load each temporary file into your editor and try to identify what the file contains, sorting out the individual pieces (each a cluster full) that belonged to each original file in turn. Your aim is rebuilding your files one at a time, saving each of them to another disc as you go.

IS IT WORTH IT?

Obviously in the worst cases this can be an extremely long-winded and tedious job, especially if you have a large number of long files to rebuild, but I never promised it would easy. Regular defragmenting of discs doesn't just have implications for disc performance, it's the best guard against this situation too, so it's doubly worth carrying out the exercise from time to time.

Obviously, if you have to do it, text files are fairly easy to recover this way because you can read the data. On the other hand, program files and non-ASCII data files can't be treated like this and even after a CHKDSK repair, many such files will probably remain a lost cause.

That's not to say that CHKDSK is a waste of time in such cases; after all, the only alternative to a CHKDSK repair, if you have a corrupted FAT or directory, is reformatting the disc for flopppies, or re-allocating the whole DOS partition on a winchester. Both of these obliterate all the files and directories on a disc, but CHKDSK doesn't, so it still has a place even if you don't intend to recover and reconstitute files manually.

As usual at this point we're out of space, but next month we'll have a little light relief before continuing the CHKDSK story.

```
 1190 DEF FNfindcall(I$):LOCAL C$,T$,A%
 1200 A%=B%:C$="":IF T%>0 REPEAT:T$=$(A%
+2):C$=$(3+LEN(T$)+A%):A%=FNnext(A%):UNT
IL T$=I$ OR A%=0:IF T$<>I$ C$=""
 1210 =C$
 1220 :
 1230 DEF PROCimport(I$,C$)
 1240 LOCAL F$,A%,F%,Z%,T$:Z%=FALSE
 1250 IF INSTR(C$,".") REPEAT:A%=B%:REPE
AT:T$=$(A%+2):F$=$(LEN(T$)+3+A%):A%=FNne
xt(A%):UNTIL T$=C$ OR A%=0:C$=F$:UNTIL I
NSTR(C$,".")=0:IF LEFT$(T$,2)="@." ENDPR
OC
 1260 F$=D$+LEFT$(I$,INSTR(I$,".")-1):I$
(0)=RIGHT$(I$,LEN(I$)-INSTR(I$,".")):IF
F$=D$+"@" ENDPROC
 1270 C$(0)=C$:i%=OPENIN F$:IF i%=0 PRIN
T'"Warning : Can't extract ";I$:PROCoutp
ut(t%,"Warning : Can't extract "+I$,FALS
E):ENDPROC
 1280 PRINT'"Searching ";F$
 1290 REPEAT:L$=FNinput(i%,FALSE)
 1300 UNTIL EOF#i% OR L$="#0"+I$(0)
 1310 IF L$<>"#0"+I$(0) PRINT"Warning :
Can't locate ";I$(0);" in ";F$:CLOSE#i%:
PROCoutput(t%,"Warning : Can't locate "+
I$(0)+" in "+F$,FALSE):ENDPROC
 1320 PRINT"Extracting ";I$(0);" (";C$(0
);")":REPEAT:L$=FNinput(i%,Z%)
 1330 F%=VAL(RIGHT$(L$,LEN(L$)-1)):IF F%
>0 I$(F%)=FNstrip(L$):C$(F%)=FNinsert(I$
(F%)):IF INSTR(C$(F%),".") REPEAT:C$(F%)
=FNfindcall(C$(F%)):UNTIL INSTR(C$(F%),"
.")=0
 1340 IF Z% AND L$<>"#<" PROCoutput(t%,L
$,FALSE)
 1350 IF LEFT$(L$,2)="#!" Z%=TRUE:PROCou
tput(t%,"--------",FALSE):PROCoutput(t%,
C$(0),FALSE):PROCoutput(t%,RIGHT$(L$,LEN
(L$)-2),FALSE)
 1360 UNTIL L$="#<"
 1370 REPEAT:L$=FNexp(FNinput(i%,TRUE)):
IF LEFT$(L$,2)<>"#>" PROCoutput(o%,L$,TR
UE)
 1380 UNTIL LEFT$(L$,2)="#>":CLOSE#i%
 1390 ENDPROC
 1400 :
 1410 DEF FNinput(c%,W%)
 1420 LOCAL Q$,I%:Q$="":IF EOF#c% THEN =
Q$
 1430 REPEAT:I%=BGET#c%:IF LEN(Q$)<255 A
ND I%<>13 AND (I%<>32 OR W%) Q$=Q$+CHR$(
I%):IF I%=33 AND W%=FALSE W%=TRUE
 1440 UNTIL I%=13 OR EOF#c%
 1450 =Q$
 1460 :
 1470 DEF FNstrip(I$)
 1480 LOCAL A%,B%:A%=0:REPEAT:A%=A%+1:B%
=ASC(MID$(I$,A%,1)):UNTIL A%=LEN(I$) OR
(B%<>35 AND (B%<48 OR B%>57))
 1490 IF A%=LEN(I$) =""
 1500 =RIGHT$(I$,LEN(I$)-A%+1)
 1510 :
 1520 DEF FNexp(I$):LOCAL Q$,A%,B%,T%,I%
 1530 Q$="":A%=0:I%=FALSE:IF I$="" =""
 1540 REPEAT:A%=A%+1:B%=ASC(MID$(I$,A%,1
)):IF I% OR (B%<>35) Q$=Q$+CHR$(B%):IF B
%=34 I%=NOT(I%)
 1550 IF I% AND B%=35 B%=32
 1560 IF B%=35 T%=ASC(MID$(I$,A%+1,1)):I
F T%<48 OR T%>57 Q$=Q$+CHR$(35):B%=32:IF
 T%=35 AND NOT(I%) A%=A%+1
 1570 IF B%=35 T%=VAL(RIGHT$(I$,LEN(I$)-
A%)):Q$=Q$+C$(T%):REPEAT:A%=A%+1:B%=ASC(
MID$(I$,A%,1)):UNTIL A%=LEN(I$) OR B%<48
 OR B%>57:IF B%<48 OR B%>57 A%=A%-1
 1580 UNTIL A%>=LEN(I$):=Q$
 1590 :
 1600 DEF PROCoutput(c%,I$,E%)
 1610 IF E% I$=STR$(L%)+I$:L%=L%+10
 1620 FOR A%=1 TO LEN(I$):BPUT#c%,ASC(MI
D$(I$,A%,1)):NEXT:BPUT#c%,13
 1630 ENDPROC
 1640 :
 1650 DEF PROCtitle(P$,Y%):IF (?&355)<>7
 ENDPROC
 1660 P$=CHR$(141)+P$
 1670 PROCcentre(P$,Y%):PROCcentre(P$,Y%
+1)
 1680 ENDPROC
 1690 :
 1700 DEF PROCcentre(P$,Y%)
 1710 LOCAL M%,X%:M%=?&355
 1720 IF M%=2 OR M%=5 X%=9:ELSE IF M%=0
OR M%=3 X%=39:ELSE X%=19
 1730 X%=X%-LEN(P$)DIV 2:PRINTTAB(X%,Y%)
;P$
 1740 ENDPROC
```

Please do keep sending in your hints for all BBC and Master computers. Don't forget, if your hint gets published, there's a financial reward.

Multiple EQUBs

Andrew Rowland

Like David Holton (BEEBUG Vol.11 No.1 p.38) I used to bewail the typing involved in entering several consecutive EQUBs. The short macro below makes it easy. Line 10000 is an example of its use. Note that the byte list is enclosed in quotation marks (as it is a string) and you should avoid any expressions including a comma, such as *array(2,4)* or *ASC","*.

```
10000 .eg OPT FNequb("4,&71,ASC""0"",6")
10010 :
10020 DEF FNequb(A$):LOCAL byte
10030 A$=A$+",":REPEAT byte=EVAL(A$)
10040 A$=MID$(A$,INSTR(A$,",")+1)
10050 [OPT pass:EQUB byte
10060 ]UNTIL A$="":=pass
```

REDEFINING THE UNDEFINED VARIABLE

Mr Toad

Bernard Hill (BEEBUG Vol.10 No.9 p.57) noted the useful fact that a variable can be incremented without previously having been declared: x=x+1 doesn't give an error even if it is the first reference to the variable x. Furthermore, you can declare a variable to be itself: x=x also gives no error if x hasn't been declared, and initially sets x to zero, avoiding the need for an extra line to do just this. On subsequent executions of this line, the statement changes nothing. Thus the following code will work:

```
REPEAT
x=x
PRINT x
x=x+1
UNTIL x=5
```

PROGRAM PROTECTION

Andrew Armstrong

One good way of protecting a Basic program from being listed is to stop Basic from finding the end of program in memory. A Basic program has a character code 255 at the end of it to signify the end of the file. If Basic cannot find this byte then it reports the 'Bad program' error message. To protect your program, load it in and type:

```
PRINT ~PAGE
PRINT ~TOP
?(TOP-1)=0
*SAVE name AAAA BBBB
```

where AAAA is the current value of PAGE (the first hexadecimal value printed) and BBBB is the value of TOP, as returned by the second PRINT statement. The program can then be loaded in at any time and RUN in the normal way, but any attempt to list the program will produce the 'Bad program' message.

THE CASE OF INPUT

Jonathan Temple

When you want to input a character and ensure that the resulting ASCII character is the upper case equivalent of the key pressed, use:

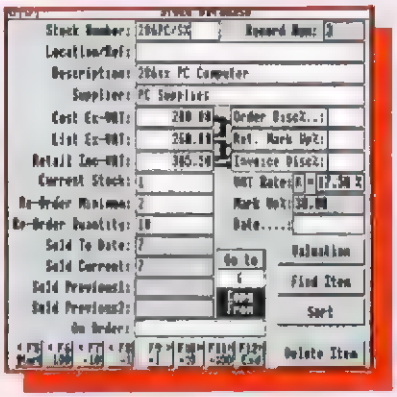
```
G=GET AND &DF
```

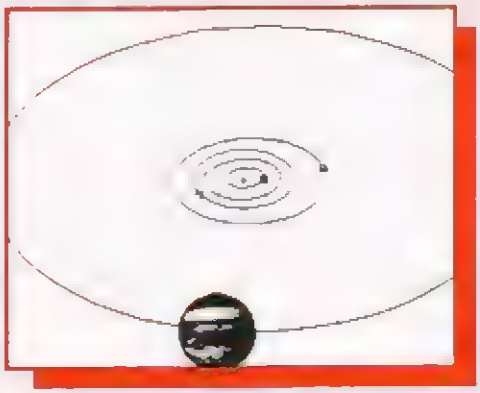
Thus, if you press 'x', the value in G will be the ASCII value of 'X'. moreover, if you use:

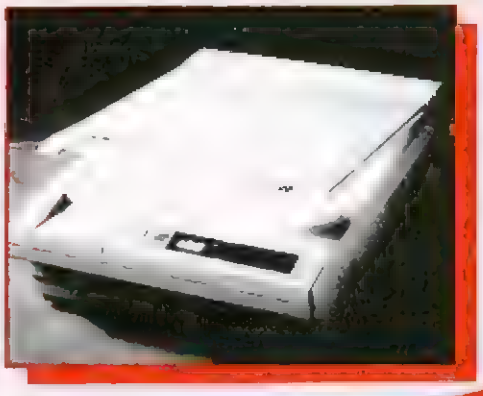
```
G=GET AND &CF
```

when pressing any number key (0 to 9), then the resulting code will be converted to the key's numeric value. So pressing the '2' key, for example, will set the subsequent value of the variable G to 2. note that pressing ant key outside the implied range can give unexpected results.

RISC USER







The Archimedes Magazine & Support Group

RISC User continues to enjoy the largest circulation of any subscription magazine devoted solely to the Acorn Archimedes range of computers. Its in-depth, authorative approach appeals to all users, whatever their interest and level of expertise, and the lively mixture of articles, programs, reviews and news is carefully selected to cater for all Archimedes owners from beginner to expert.

Existing BEEBUG members, who want to find out more about the Archimedes range, may either transfer their existing subscription to RISC User (at no extra charge), or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations, and the latest information from Acorn and other developers for both the BBC micro and the Archimedes range. RISC User is the magazine for enthusiasts and professionals at all levels.

The Archimedes Magazine & Support Group

Here are some articles and series published in the most recent issues of RISC User:

ACORN'S A4 NOTEBOOK
A comprehensive full colour supplement devoted to the new laptop from Acorn.

VIRTUAL REALITY EXPLORED
An entertaining article about the modern subject of virtual reality and a program exploring 3D images on the Archimedes.

AUTO-BOOTING THE DESKTOP
An article exploring ways of booting your system when compression files are used.

PROPHET
A look at the latest accountancy package from Apricot Studios.

PC PAGES
A regular series devoted to the PC emulator and to the running of PC software on the Archimedes.

ORRERY
A review of this simulation package from Spacetech which claims to bring the planetarium into your home.

COLOUR SCANNING AT LARGE
A look at Iriam's A4 colour scanner.

WIMP FUNCTION/PROCEDURE LIBRARY
A collection of functions and procedures (in four parts) providing all the necessary building blocks for your own Wimp programs.

WRITE-BACK
New readers' section of RISC User for comment, information, help - a magazine version of a bulletin board.

USING ANSI C
A series of articles on programming Desktop applications in C.

WP/DTP
A regular column on using different DTP and WP packages.

INTO THE ARC
A regular series for beginners.

TECHNICAL QUERIES
A regular column attempting to answer your technical queries.

FREE TEXT DATABASE

I was interested to receive G.T.Swains' communication on using Wordwise Plus as a free text database (see Mr.Pope's letter in Postbag, Vol.10 No.9). It had already occurred to me to write my own segment program in WW+, but before embarking on that I thought it better to see if anyone else had already done so, and what features they had found desirable to include. A further letter from Robert Clayton drew my attention to TexBase (see BEEBUG Vol.10 Nos.4, 5 & 6), and I am studying that. It certainly seems to have much that I could use, but its WYSIWYG format may be a drawback for the small TV I use.

My thoughts are crystallising along the lines of a database program which will call separate pieces of text into a a WW+ segment. I do not think it would be too difficult to write a WW+ database, but the labour is not welcome if I can find what I need ready made.

John Pope

As a result of the response to Mr.Pope's original letter, we are publishing in this issue an update to TexBase from another reader, Mr.H.Javes, and in the future an article by Mr.Swain on using a WW+ segment program.

SUPPORT FOR ELECTRON USERS

Since the sad demise of Electron User (published by Europress Ltd.) we Electron users have been left a little in the lurch. There is still the Electron User Group, 134 Great Knightleys, Basildon SS15 5HQ. This is doing a great job.

As the Electron is a close relative of the Beeb, perhaps you could take Electron Users under your wing as well. I am sure there is much hardware and software interchangeability between the two. I recently fitted a Watford View Printer Driver Pal-ROM, which is for the Beeb, and lo and behold it works just fine.

Terry Monaghan

As Mr.Monaghan says there are few differences between Beeb and Electron. The latter has no mode 7, for example, but any programs using this mode can be easily converted. Other differences do exist however, and users should always check carefully if they choose to purchase Beeb hardware or software for an Electron. We will not be including any specific support for the Electron, but much of what we publish is relevant, and BEEBUG is the only magazine devoted solely to the 8-bit BBC micro series.

512 FORUM BOOSTS MORALE

When, from the calm backwaters of the Beeb, one looks out at the fast flowing turbulence of the PC market, one can surely be forgiven for having twinges of doubt and anxiety. PCs are a great temptation with their ever higher speeds, enormous memories and prices dropping like a stone.

Robin Burton's 512 Forum in Vol.10 No.10 helped to clarify the PC situation, and provided what I have felt I needed for some time: morale boosting assurance that I am not foolish to be hanging on to my trusty M128.

Duncan Horne

It is always nice to be appreciated, and Duncan Horne's comments confirm just how much life and use is still left in any BBC micro if you are unconvinced by the latest trendy equipment.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

EXT 90/12 Gould switched mode PSU £55, Toolbox 2 £8, Just a Mot (cassette covering vocabulary French) £6, French Mistress (French language learning aid) £5, Shadow £5, Multi-Forth 83 Eprom £26, Edikit Eprom £7, Spellcheck III Eprom £17, Toolkit Eprom £7, Tape to Disc Eprom £7, Print Master Eprom £20, Spellcheck discs £10, Database disc £7, Forth Language £3, Romit Eprom £17, Printwise disc £20, Quickcalc disc £10, Overview Eproms £60, Master ROM cartridges £5, Care ROM cartridge Quad £7, 2764 Eproms (new) £2.50 each, 2732a Eproms (new) £3.25 each, 6264 LP memory (new) £3.75 each. Software supplied with manuals and original packages. Tel.(0254) 706127.

M128 with fitted 512 co-processor, Gem, word and spreadsheet all fitted and included as well as graphics, screen print and other facilities. Two Cumana disc drives and Zenith monochrome green VDU and all necessary cables and pair joysticks, original boxes, all manuals and reference books as well as sundry books on programming in basic etc. for sale as a complete lot for the bargain price of £550. Tel. (0626) 62782.

WANTED: ATPL sideways ROM board for B+, Gem software, Artline, Gem Draw, Gem Scan, Gem DTP, Mastering DOS Plus book. Tel. (0621) 815162.

WANTED: ATPL sideways ROM board for B+, Socinisk PC+. Tel. (0621) 815162.

Swap: Viewstore and Viewspell ROM and 5.25" utility discs for equivalent 3.5" Master compact version. Also want LISP manual "LISP on the BBC microcomputer and Acorn Electron" Tel. (0423) 884069 eves.

Microvitec Cub 14" colour monitor £100 o.n.o. Viglen 5.25" 40/80 DS with 25 discs in lockable box £75 o.n.o. Computech Integra B expansion board 64k SRAM battery backed, RTC etc. £75, o.n.o. View 2.1 ROM with manuals £10, Comal ROM with manual £10, C.C graphics extension ROM with manual £10, BEEBUG vols. 9&10 £5, Elite 5.25" with novel and poster £5, E-Type 5.25" £5, BBC B issue 3, no disc interface, useful for spares and repairs £30 o.n.o. Tel. (0705) 737840 eves and w/ends.

Epson LQ800 24 pin printer, tractor unit, parallel and serial ports, cables and manuals, boxed £160, 512 co-processor with mouse DOS+ 2.1 £100, Olivetti ET55 electronic typewriter, spare daisywheel, boxed £70, ESM colour screenprint ROM for Star printers £20 all in excellent condition. Tel. (0900) 68231.

Master 128/512 (with user guides), Canon PW 1080A printer (with manual), Philips CM8553 colour monitor (green screen button), Akhter double /switchable disc drive (with manual), ROMS; Interword, Interbase, Spellmaster, Master ROM, MOS-Plus, Ample 5000 with Nucleus ROM, Nidd Valley Digimouse, Pair Twilstar Printer stands, 2 disc boxes, Watford Electronics disc holder, double ROM cartridge, 2x2 switchable ROM cartridges. Utility software; Master Welcome & Utilities disc, Akhter BBC disc, Micro User Mini Office, Inter-Base examples, Spell-Master programs, Discmaster, Canon Utilities disc. 128 software; Micro-Aid Family History system, Help and Help ROM Image, Mewsoft Fax-File extension. 512 Software; Pipedream, Pipedream spell-check, Dabhand Master 512 User guide disc, Master Shareware collection. Various books and games, reasonable offer accepted for the complete system. Tel. (0983) 874282, leave name and address and details will be sent.

A3000 4Mb RAM (just over a year old), Ultimum podule racking system, Acorn monitor, 20Mb hard drive, full set of manuals and software worth over £1000 (all new versions and recent publications) £1400. Tel. (0532) 584857 after 6pm.

Mexican-built BBC B with View, Speech, NFS, 40/80T disc drive, Cub colour monitor, bound manuals, joystick as new £295 may split, CST Procyon IEEE interface £15, 9" NEC mono monitor £25, mono A310 with 20Mb hard disc and keyboard £475, Wordwise ROM £12, ATPL ROM board £20, mouse as new £30, 40/80T switchable TEAC drive £50. WANTED: Starword ROM and Electron joystick interface software. Tel. (0483) 480632.

Solidisk DFS and ADFS interface kit with instruction manuals £20. Tel. (0732) 863105.

BBC B issue 4, Opus 40/80T DD with own PSU, Microvitec medium resolution colour monitor, AMX mouse, joysticks, AMX Pagemaker, AMX Super-Art, View, Viewstore, Viewsheet, Database, User Guides, 30 Hour Basic (book), Elite £300. Tel. (0249) 816463.

BEEBUG Tenth Anniversary Competition

To celebrate our tenth summer of publication, we're giving away a copy of our Tenth Anniversary Disc (see page 46) to each of 10 lucky readers. Simply complete the puzzle below and send it (or a photocopy if you don't want to deface your magazine) to BEEBUG Competition, RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Entries should reach us by Monday 17th August, and the first ten correct entries we pull out of the hat will receive their prizes. Don't forget to state whether you want a 5.25" DFS or a 3.5" ADFS disc.

Note: If you have already bought the 10th Anniversary Disc, you may claim any other BEEBUG magazine disc as your prize, except MagScan (i.e. you may have Applications I or II, General Utilities, Arcade Games, Board Games, or ASTAAD, or any magazine disc). If this is the case, please make this clear on your entry form.

BEEBUG WORDSEARCH

To complete the puzzle, search for each of the words listed below in the letter grid. Words can be horizontal, vertical or diagonal and can be spelt in any direction (though always in a straight line). You should mark each word on the grid with a line.

As an extra challenge, you must also find in the grid a word that is not listed below, and as a clue the word is the name of one of Acorn's 8-bit computers.

Form cut from here

BEEBUG Tenth Anniversary Competition

To celebrate our tenth summer of publication, we're giving away a copy of our Tenth Anniversary Disc (see page 46) to each of 10 lucky readers. Simply complete the puzzle below and send it (or a photocopy if you don't want to deface your magazine) to BEEBUG Competition, RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Entries should reach us by Monday 17th August, and the first ten correct entries we pull out of the hat will receive their prizes. Don't forget to state whether you want a 5.25" DFS or a 3.5" ADFS disc.

Note: If you have already bought the 10th Anniversary Disc, you may claim any other BEEBUG magazine disc as your prize, except MagScan (i.e. you may have Applications I or II, General Utilities, Arcade Games, Board Games, or ASTAAD, or any magazine disc). If this is the case, please make this clear on your entry form.

BEEBUG WORDSEARCH

To complete the puzzle, search for each of the words listed below in the letter grid. Words can be horizontal, vertical or diagonal and can be spelt in any direction (though always in a straight line). You should mark each word on the grid with a line.

As an extra challenge, you must also find in the grid a word that is not listed below, and as a clue the word is the name of one of Acorn's 8-bit computers.

Duplicated page in scan. Unknown reason

Magazine Disc

JULY 1992
DISC CONTENTS

CORKSCREW - This disc contains the Basic library manager programs as well as the two example libraries (containing routines for calendar calculations and dynamic memory handling) which are described at the end of the article in the magazine.

HIGH SECURITY CIPHERS (PART 1) - The Sphinx and Sesame programs from the first part of this encryption article.

LOGO AND THE FAMILY TREE - Five LOGO programs are included on this disc, which implement the ideas discussed in the magazine. There is also a text file that explains how to use the programs with a LOGO system.

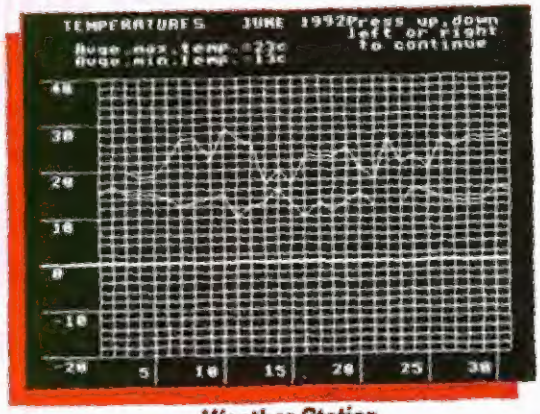
ROM CONTROLLER - A program to help you keep track of the sideways ROMs in your machine. It allows you to see the status of all your ROMs and sideways RAM banks, and to save ROM images to disc easily.

BEEBUG WORKSHOP: LINKED LISTS - The program on this disc implements the data structure of linked lists. Using this program you can enter and delete data,search for specific data and display all the data in the list, all using linked lists.

WEATHER STATION (PART 2) - This program displays the air temperature for January 1988 in the author's home town. Your own data can be added to display thermometer readings for your own area.

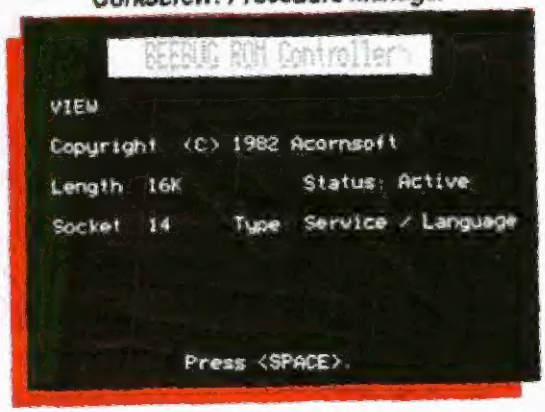
TEXBASE UPDATE - The disc contains an updated version of the TexBase program Search, which should be used with the complete TexBase system (as published on the BEEBUG magazine disc for 10:6).

MAGSCAN DATA - Bibliography for this issue(Vol.11 No.3).



Weather Station

Corkscrew: Procedure Manager



ROM Controller

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50p FOR EACH ADDITIONAL ITEM)

Back issues (5.25" and 3.5" discs from Vol 5 No 1) available at the same prices

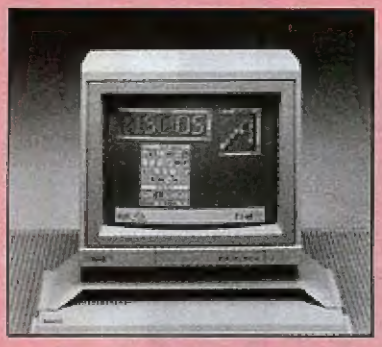
DISC (5.25" or 3.5") SUBSCRIPTION RATES	UK ONLY	OVERSEAS
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

Prices are inclusive of VAT and postage as applicable. Sterling only please

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

Upgrading to Archimedes? We've got everything you need...

The BBC A3000



The A3000 is the entry level Archimedes with the powerful multi-tasking window operating system (RISC OS).

Supplied with **1Mb RAM, internal 3.5" drive, mouse, software (Paint, Draw, Edit, Maestro music and various games)** as well as stereo sound, this really is an exciting machine with a long future ahead of it. It is very simple to use as everything you will want to do is shown on the screen in pictorial form (in a window).

The computer's memory may be expanded to a massive 4 Mb and hard drives and various interfaces may be fitted. A colour monitor will be needed as the A3000 cannot be used with a TV on its own.

A3000 Learning Curve

This system will be of particular interest to anyone looking for a general system to combine home use with education or business.

In addition to the A3000 and the software supplied with it, the Learning Curve includes:

- 1st Word Plus Word Processor with built-in 70,000 word spelling checker
- Genesis Plus - an integrated information system
- PC Emulator and DOS 5.0
- 120 minute Audio Training Tape
- Pacmania game
- Monitor Plinth (with colour systems only)

A3000 Special Access Systems

The A3000 Special Access is a version of the computer fitted with the Serial Upgrade and Morley's User/Analogue expansion card. Also included is a disc of utilities to improve access to the computer by physically disabled and visually impaired users and a Special Needs computing handbook. Prices shown include our special offer of 1 Mb RAM.

BEEBUG A3000 DTP System

- A3000 Computer
- Acorn Colour Monitor & Plinth
- 2 Mb RAM
- 20 Mb Internal Hard Drive
- Ovation DTP
- Optional User Port

This special offer provides an excellent system ready for immediate use. The hard drive, RAM and Ovation DTP are already installed so that you can simply turn on and start. A User Port is also included but requires an optional Power Supply unit.

DTP System Options

1) With a 40 Mb drive instead of the 20 Mb
2) Including the Learning Curve
3) With a User Port

Should I consider an A3000 or an A5000?

The A3000 and the A5000 will run the same programs. If you are likely to want hard drives and to use the computer a lot, then you should consider the A5000. If you are unsure of your requirements and of whether your use will justify the extra expense, then the A3000 is probably best, you can always upgrade it later.

Archimedes A5000

The A5000 is Acorn's latest 32 bit computer, housed in a solid metal casing with mouse and separate professional PC AT style keyboard. It features RISC OS 3, the new version of the popular multi-tasking window operating system used on the Archimedes. The A5000 comes complete with various items of software built in, including **Paint, Draw, Edit, Maestro** music and games.

The A5000 offers **2Mb RAM** as standard (which can be upgraded to 4Mb), an internal **40Mb IDE hard drive**, 3.5" high capacity (1.4 Mb) floppy drive and an **ARM3 processor** to make the machine run at an incredible speed. Expansion is available via the 4 slot backplane and included in the price is an excellent multi-scan colour monitor. This really is a stunning machine at a very keen price.

A5000 Learning Curve

The A5000 Learning Curve combines an excellent computer with a great collection of software to provide a good platform for home and business users.

As well as the standard Applications Suite (Edit, Paint, Draw etc) supplied with the system the A5000 Learning Curve also includes the following:

- PC Emulator and DR DOS 5.0. This will enable you to run most DOS programs on the A5000 at an acceptable speed
- 1st Word Plus wordprocessor with 70,000 word built-in spelling checker
- Acorn DTP. A Desk Top Publishing package to combine text and graphics to produce professional documents
- Audio Training Tape lasting 120 minutes to familiarise you with the computer
- Pacmania game
- Genesis Plus integrated information system
- Parents Guide to the National Curriculum Archimedes

Acorn Inkjet Printer

The A3000 and A5000 Learning Curve are also available with Acorn's 300 DPI Inkjet printer at a very competitive price. Combining quiet printing with 'near laser' quality, this is an option worth considering.

Code	Product	ex VAT	Code	Product	ex VAT
0255g	A3000 Entry System	599.00	0204g	A3000 DTP System 40 Mb	1149.00
0256g	A3000 Colour System	798.95	0258g	Learning Curve DTP 20 Mb	1039.00
0220g	A3000 Learning Curve Entry	637.45	0205g	Learning Curve DTP 40 Mb	1189.00
0221g	A3000 Learning Curve Colour	850.21	0333g	Acorn Ink Jet Printer	234.89
0257g	A3000 DTP System 20 Mb	999.00	0211g	A5000 Computer & Colour Mon.	1499.00
5297b	User Port PSU	17.00	0213g	A5000 Learning Curve Colour	1531.06
0297g	A3000 Special Access	679.00	0194g	Archimedes A540 Computer	2495.00
0299g	A3000 S.A. Colour + Press Stand	899.00	0195g	Archimedes A540 Colour (not multi-sync)	2694.00

BEEBUG Ltd • 117 Hatfield Road, St. Albans, Herts AL1 4JS • Tel 0727 40303 • Fax 0727 860263